Chromatics

CGC 7900

SERIAL PORT CONTROLLER MANUAL

© 1980 Chromatics

SERIAL PORT CONTROLLER MANUAL

CGC 7900 SERIES
COLOR GRAPHICS COMPUTERS

CHROMATICS
CGC 7900 Series

Serial Port Controller
Application Guide

CHROMATICS

CGC 7900 SERIES COLOR GRAPHICS COMPUTER SYSTEM

SERIAL PORT CONTROLLER (SPC)
APPLICATION GUIDE

Copyright (c) 1982 by Chromatics, Inc.
2558 Mountain Industrial Boulevard
Tucker, Georgia 30084

Phone (404) 493-7000
TWX 810-766-8099

May, 1982

INTRODUCTION

This Application Guide describes the Serial Port Controller, or SPC, an optional card in Chromatics' CGC 7900 series. The SPC is designed to handle low-level data communications chores in the 7900 system, for up to 4 RS-232 ports.

By relieving the main CPU of the burden of handshaking and buffering, the SPC can greatly enhance system throughput. The SPC contains its own Z80 processor, and firmware which runs the normal "read character, write character" operations. This Application Guide is intended for the user who wants to customize SPC firmware for special purposes. We will discuss the SPC architecture, and provide programming examples.

This document is written for the experienced programmer. The SPC firmware is written in Z80 assembly language, and you will need access to a Z80 assembler and development system, or a compiler capable of generating Z80 code. For high-speed applications, running one or more ports at high baud rates, you will probably have to write the majority of the firmware directly in assembly language for efficiency. Current SPC firmware was developed on Chromatics' CG series color graphic computer systems.

We will begin by describing the SPC, its architecture, and how it operates in a standard CGC 7900 system. From there, we will proceed to the advanced features of the hardware, including some which are not normally used (but are installed). These features include the ability to run one or more ports with external clocks; daisy-chaining up to four SPC boards in a system; and interrupt-driven I/O.

Other CGC 7900 documentation available from Chromatics includes the CGC 7900 User's Manual, OEM Manual, and Disk Operating System Manual. Additional SPC documentation includes the circuit descriptions, test procedures, schematics, and source listing for the firmware. Some of this documentation is considered proprietary, and you may be required to file a non-disclosure agreement.

ARCHITECTURE

The SPC consists of a Z80 processor running at 2.5 MHz, two 2532-type EPROMs for onboard firmware (up to 8K bytes), 4K bytes of onboard RAM, and 1K bytes of two-port RAM. The four serial ports are each handled by a 2661 Enhanced Programmable Communications Interface (a friendly USART). The remainder of the circuitry is "glue" logic which holds the system together, and provides interrupts, interprocessor signalling, and i/o decoding.

The SPC is, in fact, a complete Z80 computer system. It only requires a few signals from the 7900 bus in order to function. This means that the SPC can continue to run during times when the main 68000 system processor is unable to operate (during DMA transfers, for example). Within the 4K of onboard memory, each port has two buffers (transmit and receive), each of which is 256 bytes long. A substantial amount of data can be buffered in this onboard RAM before 68000 intervention is required.

The memory map on the following page details the SPC memory allocation. Onboard EPROM occupies 0000 through 1FFF hex. Two-port RAM is from 2000 to 23FF. All i/o is memory-mapped, and resides from 3000 to 301F. This includes the 2661 chips and the Flags (used for signalling between processors). Onboard RAM is located at 7000 through 7FFF. Most of onboard RAM is available for buffer area, although a small amount must be used for the Z80 stack and temporary data storage areas.

From the 68000 point of view, the SPC occupies 2K of address space, from FF0000 to FF07FF. All odd-numbered bytes in this range are mapped into the two-port RAM. Even bytes are used to access the Flags. Since the SPC is an 8-bit system, all 68000 programs talking to it must use 8-bit (byte) operations only. Using word or long word instructions will cause unhappy behavior.

The 68000 can only address the two-port RAM and the Flags. It cannot access other items in the SPC address space directly; that is the Z80's job.

The two-port RAM (known as TRAM) is the method by which the two processors communicate their needs. The firmware contains a set of commands which can be passed to the Z80 in TRAM; the Z80 will act on these commands and return results to the 68000. The command set is described in a later section of this document.

Current SPC firmware occupies only the first of the two EPROM sockets. The second EPROM is available for user-written firmware <u>at this time</u>. Chromatics reserves the right to expand SPC functions and utilize the second EPROM at some future date... swim at your own risk.

MEMORY MAP

- Z80 Side -

0000 - 0FFF	EPROM #0
1000 - 1FFF	EPROM #1
2000 - 23FF	Two-port RAM
3000 - 301F	I/O space: USARTs, flags
7000 - 7FFF	Onboard RAM

I/O space is allocated as follows:

3000	Port 0 data
3001	Port 0 status
3002	Port 0 mode
3003	Port 0 command
3004 - 3007	Port 1 (as above)
3008 - 300B	Port 2
300C - 300F	Port 3
3010	RTC reset
3018	Flag 1 (read examines, write SETs)
3019	Flag 2 (read examines, write CLEARs)
301C	Flag 3 (read or write interrupt's 68000)

- 68000 Side -

FF0001 - FF07FF	(odd bytes only) Two-port RAM
FF0000	Flag 1 (read examines, write CLEARs)
FF0002	Flag 2 (write SETs Z80 interrupt)
FF0004	Flag 4 (write RESETs Z80, read allows Z80 to run)
FF0006	Flag 3 (write CLEARs 68000 interrupt)

NOTE: Due to redundant addressing, some items also appear at addresses other than those listed above. For example, on the Z80 side, TRAM also appears at 2400-27FF, 2800-2BFF, and 2C00-2FFF. On the 68000 side, TRAM is uniquely addressed, but the Flags are not. Therefore, programmers should be careful not to access any addresses other than those listed above.

Since the SPC is inherently an 8-bit device, all 68000 programs using the SPC should use byte instructions only. Using 16-bit or 32-bit instructions will access the TRAM and the Flags simultaneously, causing strange results. **Be especially careful when accessing the two-port RAM:** the program must read a byte, skip over a byte, and read the next byte from the next odd address. A sample program fragment might be:

```
        LEA     TRAM,A0         ;A0 -> 2-port RAM
Loop    MOVE.B  (A0)+,(A1)+     ;copy one byte from TRAM
        ADDQ.L  #1,A0           ;skip odd bytes
        DBRA    D0,Loop         ;continue
```

NOTE: Any time the Z80 accesses Flag 3, whether during a read OR a write, it will set an interrupt to the 68000. Be careful when examining memory in the I/O space, since reading it can cause unwanted interrupts.

FIRMWARE

The SPC firmware operates with the 7900 Terminal Emulator, TERMEM. Version 2 firmware, and later versions, also support the Idris operating system. Interaction with Idris is more complex than TERMEM, and we will describe TERMEM first.

It is TERMEM's purpose in life to read characters from logical devices and write them to other logical devices. A device assignment structure allows each logical device to be "connected" to one or more physical devices. Each of the ports on the SPC is considered to be one physical device, assignable for input, output, or both.

When TERMEM is running with the SPC, two basic operations are possible: write a character to a port, and read a character from a port. TERMEM operates on a character-at-a-time basis, so more complex interactions are not required. A third operation "reconfigures" a port, setting up baud rate, handshaking, and character format. Each of these is discussed in the "Commands" section of this document.

When the SPC is powered-up or reset, the Z80 begins executing code from its onboard EPROM. It initializes the four serial ports with default parameters, and enters a simple loop which performs these functions:

```
Service port 0.
Service port 1.
Service port 2.
Service port 3.
Check for commands from the 68000, and process
them if necessary.
```

To service any port, the Z80 reads the port status from the 2661 chip. If a character has been received, it is loaded into onboard RAM. If a buffer becomes full, the proper handshaking protocol is performed. Then the transmitter side of the port is serviced. If a port shows "transmitter ready," a character is pulled from onboard RAM and transmitted. The firmware also transmits and times a "break" pulse if requested. Sending an FF hex to any port will generate the break pulse.

To simplify interaction with TERMEM, all SPC operation is in polled mode - no interrupts are used. Since the SPC can asynchronously buffer all transmitted and received data, there is no need for interrupts under TERMEM. A sample exchange between the processors might be as follows:

68000	Z80
Wait for Flags to signal that the TRAM is available for a command.	
Put "read character" command into TRAM, along with the port number.	
Release TRAM to the Z80.	
	Recognize that TRAM is available, and read the command.
	Read a character from the appropriate buffer.
	Put the character into TRAM and send it away.
Wait for TRAM to return.	
Read the character from TRAM.	
Process the character.	

This is the basic method of operation for all transactions between the Z80 and the 68000. The Flags are used to synchronize TRAM accesses, and also to allow each processor to interrupt the other. These Flags are discussed next.

FLAGS

Any multiprocessor system must use some form of signal between the processors, to insure orderly transfer of data. The SPC uses a set of semaphores, or Flags, which may be tested or set under various conditions. The Flags are actually hardware flip-flops which can each store one bit of information. When you read a Flag, its information appears on the high bit of the byte (bit 7). This allows a simple "branch-if-minus" instruction to act on the state of the flag.

Flag 1 is a semaphore which controls access to the two-port RAM (TRAM). When Flag 1 is SET, the 68000 owns the TRAM. When it is CLEAR, the Z80 owns the TRAM. Each processor may give up the TRAM by writing to Flag 1, but may not "grab" the TRAM. Each processor must wait for TRAM access by testing the state of Flag 1. Other Flags, discussed below, are used to request ownership of the TRAM. Flag 1 is CLEARED after a reset.

Flag 2 is used by the 68000 to send a signal to the Z80. The 68000 sets this Flag by writing to it. The Z80 can test this Flag by reading it. Alternatively, if the Z80 has enabled interrupts, Flag 2 will interrupt the Z80. In either case, the Z80 will write to Flag 2 to clear it. The 68000 cannot read back the state of Flag 2, so some other means must be used to tell whether the Flag 2 signal has been serviced. (This can be accomplished by Flag 1 or Flag 3.) Flag 2 is CLEARED after a reset.

Flag 3 is used by the Z80 to interrupt the 68000. When the Z80 writes or reads Flag 3, logic on the SPC requests an interrupt of the specified priority (set by switches on the SPC card). The interrupt will be acknowledged by the 68000 when its execution priority drops below the SPC's request priority. The 68000 clears this interrupt by writing to Flag 3. Since the Z80 cannot read back the state of Flag 3, one of the other Flags must be used to tell whether the interrupt has been serviced. Flag 3 is CLEARED after a reset.

Flag 4 allows the 68000 to reset the SPC board. This is equivalent to a hardware reset signal, and causes the SPC to clear all Flags and begin executing onboard firmware at address zero. It allows the 68000 to bring the Z80 to a known state, without resetting any other system hardware. Writing to Flag 4 resets the SPC and holds it in a reset state. Reading from Flag 4 allows the Z80 to run. Note that the Z80's firmware initializes some onboard RAM locations and sets up the USARTs; this may interfere with RAM-resident Z80 programs. Flag 4 is CLEARED by a system reset, but a system reset pulse will also reset the SPC.

Examples of Flag usage (68000 side):

```
FLAG1   EQU     $FF0000         ;Equates for Flags
FLAG2   EQU     $FF0002
FLAG3   EQU     $FF0006
FLAG4   EQU     $FF0004

Wait    BTST    #7,FLAG1        ;This loop waits
        BEQ.S   Wait            ;  for TRAM access.

Away    CLR.B   FLAG1           ;Release TRAM to Z80.

HeyYou  CLR.B   FLAG2           ;Holler at Z80

Clrint  CLR.B   FLAG3           ;Clear 68000 interrupt

Reset   CLR.B   FLAG4           ;Reset the Z80,
        TST.B   FLAG4           ;  then let it run.
```

Examples of Flag usage (Z80 side):

```
FLAG1   EQU     3018H           ;Equates for Flags
FLAG2   EQU     3019H
FLAG3   EQU     301CH

Wait    LD      A,(FLAG1)       ;This loop waits
        OR      A               ;  for TRAM access.
        JP      M,Wait          ;  (loop if minus)

Away    LD      (FLAG1),A       ;Release TRAM to 68000.

Poll    LD      A,(FLAG2)       ;Check Flag 2
        OR      A
        JP      M,IsSet         ;  and jump if set

Clrint  LD      (FLAG2),A       ;Clear Flag 2

HeyYou  LD      (FLAG3),A       ;Interrupt the 68000
```

INTERRUPTS

The SPC allows operation in polled or interrupt-driven modes. Standard firmware in the SPC, when operating with TERMEM (the 7900 Terminal Emulator program), uses only polled mode. In this mode, each processor examines the Flags to determine the status of the two-port RAM, and acts according to this status. The two-port RAM is used to pass commands and data between the two processors.

In some applications, greater system throughput is achieved by letting the SPC interrupt the 68000 when it requires service. This interrupt-driven mode of operation is effective whenever the 68000 is busy with other tasks; for example, running an operating system or applications program.

Three interrupts exist in the SPC. The first is a real-time clock interrupt, which is tied to the NMI (non-maskable interrupt) input of the Z80. This interrupt is set every 60th of a second, by the vertical retrace signal in the 7900. (Systems running on 50 Hz power will receive 50 Hz interrupts).

The clock interrupt must be cleared by the Z80 before it can occur again. The Z80 clears this interrupt by accessing address 3010 hex.

The following code is extracted from version 1 of the standard firmware, and is executed every "tick" of the real-time clock:

```
VERT    EQU     3010H   ;addr to reset int
TIME    EQU     7FFEH   ;clock bytes

Tick    LD      (VERT),A        ;clear the NMI
        EXX                     ;flip to alt. regs
        LD      HL,(TIME)       ;bump clock
        INC     HL
        LD      (TIME),HL
        EXX
        RETN
```

Several things are important about this code. Note that the upper two bytes of onboard RAM are used as a 16-bit counter, incremented every 60th of a second. Also, remember that the Z80 will always do a CALL to address 0066 hex when an NMI occurs, so this code must live at 0066, which is in the first EPROM. We use alternate register pair HL' in this service routine, which precludes use of the alternate registers anywhere else (NMI's cannot be disabled). Finally, notice that a RETN is used to end the routine. RETN restores the Z80 maskable interrupts to the state they were in before the NMI occurred.

The firmware uses this clock interrupt to time the length of a generated "break" signal. It is customary to assert "break" for about 200 milliseconds, or 12 ticks.

Version 2 of firmware includes a more complex clock service routine, for support of the Idris device drivers. In addition to the functions above, it can also execute a "wakeup" task after a certain number of clock ticks. This wakeup task is used in Idris to periodically interrupt the operating system and request SPC service.

```
VERT    EQU     3010H   ;addr to reset int
FLAG2   EQU     3019H   ;F2 address
WAKEUP  EQU     7FF7H   ;what to do when awakened
WAKTIM  EQU     7FF6H   ;running counter
OFTEN   EQU     7FF5H   ;how often to wake
TIME    EQU     7FFEH   ;clock bytes

Nmi     LD      (VERT),A        ;clear the NMI
        PUSH    HL
        PUSH    AF              ;save regs
        LD      HL,(TIME)       ;bump clock
        INC     HL
        LD      (TIME),HL
        LD      A,(FLAG2)       ;see if F2 set
        OR      A
        JP      P,Nmi9          ;jmp/no, don't wakeup
        LD      HL,WAKTIM       ;point to wakeup timer
        DEC     (HL)            ;tick it
        JR      NZ,Nmi9-$       ;jmp if not time to go
        DEC     HL              ;point to OFTEN
        LD      A,(HL)          ;get it
        INC     HL
        LD      (HL),A          ;reload WAKTIM
        CALL    WAKEUP          ;service the clock
Nmi9    POP     AF              ;restore & return
        POP     HL
        RETN
```

If Flag 2 is not set when the clock ticks, this routine degenerates into the code from version 1, except that it doesn't use the HL' register. This allows other routines to use the alternate registers, and the Idris device driver does.

If Flag 2 is set when the clock ticks, the RAM location WAKTIM is decremented. If it goes to zero, it gets reloaded from location OFTEN. Then we call location WAKEUP, which will execute the clock-driven task. WAKEUP is three bytes long and is initialized to a jump to a RET instruction. This allows other programs to use Flag 2, and if WAKEUP is left alone, the clock servicer won't affect anything.

To make use of the clock service routine, store the "tick rate" into OFTEN, and the service routine address into WAKEUP+1:

```
        LD      A,3             ;every 3 ticks
        LD      (OFTEN),A
        LD      HL,Addr         ;whom to call
        LD      (WAKEUP+1),HL
```

This would cause the routine at "Addr" to be executed every three clock ticks, 20 times a second. Of course, Flag 2 must be set or WAKTIM won't get decremented, and WAKEUP will never get called. **WAKEUP is called at the interrupt level, so it must be fast, and must save all registers it uses.** Note that if the wakeup task is not complete by the next clock tick, it could get re-entered at the interrupt level. This almost surely leads to disaster.

- Flag 2 -

The next type of interrupt is produced by Flag 2, and is used by the 68000 to interrupt the Z80. When the 68000 writes to address FF0002, Flag 2 is set. The Z80 can poll Flag 2 if polling mode is desired, or Flag 2 can generate a maskable interrupt to the Z80. Maskable interrupts are enabled and disabled by the EI and DI instructions. (Standard firmware does not use maskable interrupts, so interrupts are always disabled.) The Z80 clears this interrupt by writing to Flag 2.

Since only one source of maskable interrupts exists (Flag 2), the SPC is designed to operate in Interrupt Mode 1 as defined in the Z80 literature. Interrupt Mode 1 provides the simplest hardware interface to the Z80. The processor enters this mode by executing the instruction

```
IM      1
```

which must be included before interrupts are enabled. (Standard firmware does this.) In Mode 1, the Z80 performs a CALL to address 0038 hex when an interrupt occurs. The maskable interrupt service routine must be located at this address. It must be terminated with the instructions

```
EI
RET
```

which re-enables interrupts and continues the previous program. Of course, the interrupt service routine must save and restore any registers it uses. (The Z80 "RETI" instruction is acceptable in place of "RET", but not necessary. In any case, the "EI" must be included to allow future interrupts.)

When a maskable interrupt occurs, the standard firmware does a jump to address 7FFB. This address is initialized to contain a jump to a RET instruction. The address of your interrupt service routine should be loaded into location 7FFC. This could be done as follows:

```
INTJP   EQU     7FFBH   ;where to go when int'd

        LD      HL,Isr          ;service routine addr
        LD      (INTJP+1),HL    ;store it
```

Now if interrupts are enabled and Flag 2 is set, the Z80 will execute code at address "Isr".

- Flag 3 -

The third type of SPC interrupt is used by the Z80, to interrupt the 68000. The Z80 can set an interrupt to the 68000 by writing or reading Flag 3. If the execution priority of the 68000 is currently below the priority of the SPC interrupt, the 68000 will begin its interrupt service routine. To clear the interrupt, the 68000 must write to address FF0006, its name for Flag 3.

```
FLAG3   EQU     $FF0006
        CLR.B   FLAG3   ;reset the int
```

The address of the 68000 interrupt service routine must be loaded into one of the interrupt vectors, usually vector number $7C, which is at address $1F0. This means that before any SPC interrupts can be fielded by the 68000, you must do the following:

```
MOVE.L  #SPCisr,$1F0    ;set the vector
```

Where "SPCisr" is the address of the SPC interrupt service routine, to be executed when the Z80 rings for service.

To minimize the amount of interrupt servicing, plan your software so that certain things are implicit. For example, when the Z80 interrupts the 68000, the 68000 should not have to wait for two-port RAM access. The Z80 should insure that Flag 1 is SET before it interrupts the 68000:

```
LD      (FLAG1),A       ;send away TRAM
LD      (FLAG3),A       ;set the int
```

The 68000 can then immediately read from two-port RAM to determine why the Z80 interrupted it.

ONBOARD RAM USAGE

The 4096 bytes of onboard RAM are allocated for i/o buffers, parameter areas for ports, stack space, and system constants. The allocation shown below is for version 2 firmware.

```
7000-7800  i/o buffers:

7000-70FF  port 0 receiver buffer
7100-71FF  port 0 transmit buffer
7200-73FF  port 1 buffers
7400-75FF  port 2 buffers
7600-77FF  port 3 buffers

7800-781F  port 0 parameter area
7820-783F  port 1 parameter area
7840-785F  port 2 parameter area
7860-787F  port 3 parameter area

7880-7DFF  expansion

7E00-7EEF  Idris coroutine stack
7EF0-7FEF  system stack

7FF0-7FFF  system RAM constants:

7FF0-7FF1  COSP: Idris stack pointer storage
7FF2-7FF3  MAINSP: main stack pointer storage
7FF4       SERV: Idris clock service flag
7FF5       OFTEN: clock service rate
7FF6       WAKTIM: running clock counter
7FF7-7FF9  WAKEUP: clock service routine
7FFA       ENABLE: port-enable byte
7FFB       INTJP: Flag 2 interrupt service routine
7FFE-7FFF  TIME: 16-bit running timer
```

Note the area from 7880 to 7DFF. This is free RAM in version 1 and 2 firmware, and can be used for loading user-written code. Bear in mind that future versions of firmware may use this RAM space; don't get too attached to it.

INITIALIZATION: After a reset, certain RAM areas are loaded by the Z80. The buffers from 7000 to 77FF are not cleared out, but their contents are ignored. Parameter areas from 7800 to 787F are copied from PROM; these include the buffer counts and pointers, handshake flags, USART initialization values, and other per-port information. The area from OFTEN through INTJP is initialized as follows:

OFTEN is set to 1.

WAKTIM is set to 0.

WAKEUP jumps to a RET instruction.

ENABLE is set to 0F hex, enabling all four ports.

INTJP jumps to an EI followed by a RET.

The area from 7E00 to 7FEF is used for stack space. The system stack pointer is initialized to 7FF0 and grows down from there. Part of the Idris code requires a separate stack, which grows down from 7EF0. About 256 bytes of space are allocated for each stack, which is probably a bit much. COSP and MAINSP each hold the value of the SP (stack pointer) during use of the other stack.

If future versions require more RAM space, it will probably be allocated as follows: simple one- or two-byte values will be allocated down from 7FF0, moving the stacks down to make room. If larger chunks of RAM are needed, they will be allocated up from 7880. Use this as a guide in plaaning your RAM allocation.

THE ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE

This section discusses the EPCI, otherwise known as the 2661 communications chip. We will point out some salient features of the chip in this section, including the basic methods of programming it (in asynchronous mode). The Appendix contains a 2661 data sheet with full programming details.

Programmers who are familiar with the 8251 USART device will be pleasantly surprised by the 2661. The 2661 is similar in function, and the signal mnemonics are familiar enough that it will be easy to learn the 2661. Yet, the 2661 eliminates many of the 8251 and 8251A headaches: most of the 2661 registers are read/write, so that the Z80 "bit" instructions work conveniently for testing and changing bits. (Indeed, the designer chose to use memory-mapped I/O in the SPC to allow use of "bit" set/reset/test instructions.)

Nearly all of the 2661's functional characteristics can be altered "on the fly." This includes the number of bits per character, parity, stop bits, and other parameters which created programming nightmares in the 8251. The 2661 has an internal baud-rate generator which provides 16 standard rates. (External clocking is available, and is discussed in a separate section of this document.)

The 2661 contains nine registers, five of which are used in asynchronous applications. Four of these are available all the time, as read/write locations. The fifth is only used during initialization (usually), but is quite easy to access during operation if necessary. The registers are:

Offset	Register
0	Data in/out
1	Status
2	Mode registers 1 and 2
3	Command

The offset lists the amount which must be added to the base address of a port, in order to access a given register of that port. It is convenient in Z80 code to use the index registers, IX and IY, to access the various registers of a port. For example:

```
LD      IX,PORT0        ;IX-> base of port
SET     1,(IX+3)        ;turn on DTR
BIT     7,(IX+1)        ;test DSR
```

All of the bits will be explained momentarily. For now, note the ease of operating the 2661 through the Z80 index registers.

Register Contents

The **data input/output** register is written into in order to transmit a character. Read from this register to get a received character. (Before reading or writing, you should test bits in the status register to insure that the 2661 is ready.)

The **status register** bits are defined below. All status register bits are similar to corresponding bits in the 8251, with two exceptions: The IR (internal reset) bit is not present, since all 2661 registers are read/write and it is never necessary to software-reset the chip. Also, bit 2 has taken on an additional meaning, as described below.

Status Register

7	6	5	4	3	2	1	0
DSR	DCD	FE/ SYN det	OR	PE/ DLE det	TxEMP/ DSchg	RxRDY	TxRDY

Bits 7 and 6 indicate the state of the corresponding modem control signals on the interface. (The DCD input is not present in the 8251. This input must be TRUE in order for the 2661 receiver to function. In the SPC, if the DCD line is unconnected, it is held in a TRUE state by a resistor.)

Bit 5 indicates FE, framing error, in asynchronous mode. It is set when a character does not contain a valid stop bit. This may that the 2661 is receiving a "break" condition on the data line. (Break is indicated by FE occuring while a null character, 00, is present in the data register.)

Bit 4 indicates OR, overrun error. It is set when the Z80 has not read characters out of the 2661 fast enough, and data has been lost.

Bit 3 indicates PE, parity error, in asynchronous mode. The software may choose to ignore this bit, if parity checking is not required.

NOTE: Bits 3, 4, and 5 are all reset by a "reset errors" command to the command register. See the command register description.

Bit 2 is set when the transmitter is totally empty. (Being double-buffered, the transmitter can be ready without being empty. See Bit 0.) This bit is also set if either the DSR or DTR inputs has changed, indicating a possible change in the status of the communication link. This "DSchg" indicator is not present in the 8251.

Bit 1 indicates the receiver is ready, and data should be read from the data register.

Bit 0 indicates the transmitter is ready, and data should be written into the data register if any is available.

Two **mode registers** exist, MR1 and MR2. This is the only case in the 2661 where a register is not always read/writable, and we will discuss these registers in detail so you can avoid pitfalls.

Both MR1 and MR2 are accessed through the same address, which is PORT plus two (PORT being the base address of a chip). Mode register 1 is identical to the mode register in an 8251:

Mode Register 1

7 6	5	4	3 2	1 0
00=sync mode 01=1 stop bit 10=1.5 stop bits 11=2 stop bits	parity: 0=even 1=odd	 0=no 1=yes	00=5 bits 01=6 bits 10=7 bits 11=8 bits	00=sync mode 01=1X clock 10=16x clock 11=64x clock

Bits 7 and 6 define the number of stop bits per character.

Bits 5 and 4 define the parity (odd, even, or none).

Bits 3 and 2 define the number of data bits per character.

The total number of bits transmitted per character is actually the total of: stop bits, data bits, and parity (if enabled).

Bits 1 and 0 select the baud rate multiplier, or select synchronous mode if 00. Note that if the internal baud rate generator is being used (see MR2 below), the multiplier is ignored and any of the asynchronous multiplier values may be used.

Mode register 2 is unlike anything in the 8251. It controls the internal baud rate generator in the 2661, and also specifies the function of certain pins on the chip. These pins are used in external clocking applications.

The upper four bits of MR2 select internal or external clocking, synchronous or asynchronous operation, and define pins 9 and 25 on the chip, as follows:

Bits 7-4	TxC	RxC	Pin 9	Pin 25	Mode
0000	ext	ext	TxC in	RxC in	sync
0001	ext	int	TxC in	1x out	async
0010	int	ext	1x out	RxC in	sync
0011	int	int	1x out	1x out	async
0100	ext	ext	TxC in	RxC in	sync
0101	ext	int	TxC in	16x out	async
0110	int	ext	16x out	RxC in	sync
0111	int	int	16x out	16x out	async
1000	ext	ext	xsync	RxTxC in	sync
1001	ext	int	TxC in	brkdet	async
1010	int	ext	xsync	RxC in	sync
1011	int	int	1x out	brkdet	async
1100	ext	ext	xsync	RxTxC in	sync
1101	ext	int	TxC in	brkdet	async
1110	int	ext	xsync	RxC in	sync
1111	int	int	16x out	brkdet	async

Bits 7 through 4 of MR2 must be set to select the proper source (internal or external) of the baud rate clock, and the proper mode (sync or async). Pins 9 and 25 of the 2661 are not connected to anything unless jumpers are installed by the user, so their meaning need not concern us at this point.

Bits 3 through 0 of MR2 select the frequency of the internal baud rate generator. The available rates are listed below.

Bits 3-0	Baud Rate
0000	50
0001	75
0010	110
0011	134.5
0100	150
0101	200
0110	300
0111	600
1000	1050
1001	1200
1010	1800
1011	2000
1100	2400
1101	4800
1110	9600
1111	19200

Interaction between MR1 and MR2. After a hardware reset, the 2661 expects you to load the first mode register (MR1). It then expects you to load MR2. This sequence is necessary, since both registers must be loaded before the 2661 can be used. After loading both registers, the 2661 is again addressing MR1.

If you reload the mode registers after initialization, or if you want to read data from one or both mode registers, there is a way to tell which is which: the 2661 always points back to MR1 after you read from the command register. So, to be absolutely safe when accessing the mode regsters, use this procedure:

- Always read from the command register before accessing any mode regsister.

- Always read both mode registers, or write both mode registers. Your software may require a RAM copy of the mode register contents, to insure that you write proper data into both registers.

The **command register** of the 2661 is similar to the corresponding 8251 register. Bits 7 and 6 have been given additional meaning to support the 2661's self-test modes. Bit 3 is used in synchronous mode to support DLE transmission.

Command Register

7 6	5	4	3	2	1	0
00=normal 01=auto-echo 10=local loop 11=remote loop	RTS	error reset	send break/ DLE	Rx enable	DTR	Tx enable

Bits 7 and 6 are 00 for normal operation. The other modes are described in 2661 literature, and are used for self-test and loopback operations without processor intervention.

Bit 5 controls the RTS modem control signal.

Bit 4 resets the PE, FE, and OR error bits in the status register. Writing a "1" to this bit will reset the errors, and the bit will automatically return to zero:

```
SET     4,(IX+3)        ;reset errors
```

Bits 2 and 0 control the receiver and transmitter, respectively. The receiver-ready and transmitter-ready bits of the status register will not go true unless these control bits have been enabled.

Bit 1 controls the DTR modem control signal.

Bit 0 (TxEN) and bit 3 (break) perform in a friendly fashion. They do not affect any character which may be transmitting at the time of the command. These commands take effect after the current character (if any) has been completed. However, since the 2661 is double-buffered, the character being transmitted may not be the only character in the USART. If you turn off the TxEN bit, any character that has been written to the USART but has not yet begun transmission will be lost. It's best to wait for TxEMT before dropping TxEN.

Programming Example

The following code might be used to initialize a 2661.

```
LD      IX,PORT         ;IX-> 2661 chip
LD      A,(IX+3)        ;read the command register
                        ; to sync MR1/MR2
LD      (IX+2),7AH      ;MR1= 7 bits, even parity,
                        ; 1 stop bit
LD      (IX+2),0FEH     ;MR2= internal clocks,
                        ;9600 baud
LD      (IX+3),27H      ;turn on Tx, Rx, DTR, DSR
```

COMMANDS

Commands are passed to the SPC in the two-port RAM (known affectionately as TRAM). Each command consists of an opcode, which is placed in the first byte of TRAM. This is followed by one or more bytes to specify details of the transaction.

If the SPC is required to return a response for a given command, the opcode byte is left intact in TRAM, and is followed by the returning arguments. If no response is required, the opcode will be zeroed out, and other bytes in the TRAM are irrelevant. In all cases, when running under the firmware used by TERMEM, the SPC will return ownership of the TRAM to the 68000 when an operation is complete.

If an invalid opcode is passed to the SPC, or if the arguments to that opcode are invalid, the opcode will be zeroed out and ignored.

In the following charts, "Offset in TRAM" is given from the Z80 side. From the 68000 side, the offset would be doubled, since every other byte must be skipped.

Note that opcodes 7 and above did not exist in version 1 firmware. Opcode 8 can be used to test the firmware version of an SPC.

Opcode 1: Transmit Character

Offset in TRAM	Contents
0	1
1	port # (0..3)
2	character

Returns:

0	0

Opcode 1 is used by TERMEM to transmit a single character to a port. The character is buffered in onboard RAM and transmitted when possible. If the buffer is full, the SPC keeps TRAM ownership until there is room for at least one character in the buffer.

Opcode 2: Read Character

Offset in TRAM	Contents
0	2
1	port number (0..3)

Returns:

0	2
1	port number (0..3)
2	buffer count
3	character

Opcode 2 checks the count of received characters in a port's buffer. If the count is not zero, it also returns the oldest character in the buffer. TERMEM uses this and Opcode 3 (below) to read from a device.

The buffer count returned by opcode 2 is the number of characters in the buffer before the returning character was removed. If the count is 1, you are now reading the last character. If the count is zero, no characters are available and the contents of TRAM+3 are invalid.

Opcode 3: Check Port Status

Offset in TRAM	Contents
0	3
1	port number (0..3)

Returns:

0	3
1	port number (0..3)
2	buffer count
3	character snapshot

Opcode 3 is used by TERMEM to check if any characters are available from a port. It returns the count of received characters, and also a "snapshot" of the oldest character in the buffer. This allows TERMEM and DOS to check for the presence of certain characters (control-S to pause a listing, for example) without actually reading characters from the device.

No matter how many times you execute Opcode 3, the "snapshot" will always be the same character. Opcode 3 does not remove any characters from the buffer.

If you execute Opcode 3 and then Opcode 2, each will return the same character. Opcode 3 will produce a snapshot of that character, and Opcode 2 will read it again (and remove it from the buffer).

Opcode 4: Reconfigure a Port

Offset in TRAM	Contents
0	4
1...	character string

Returns:

0	0

Opcode 4 takes a literal string and parses it to gather commands. These commands reconfigure a port, setting handshaking, baud rate, number of bits, parity, and number of stop bits. The characters loaded into TRAM+1 and succeeding bytes must constitute an ASCII string of the following form:

<port>, <hand>, <baud>, <bits> <par> <stop>

<port> is a decimal number, 0 to 3, delimited by a comma.

<hand> is also a decimal number, delimited by a comma. It sets the port's handshake parameters as follows:

<hand>	Effect
0	No handshaking
1	Software (Xon,Xoff) Protocol
2	Hardware (DTR,DSR) Protocol
3	Both SW and HW Protocols

<baud> is a decimal number, delimited by a comma, which must be one of the following legal baud rates: 50, 75, 110, 134, 150, 200, 300, 600, 1050, 1200, 1800, 2000, 2400, 4800, 9600, 19200. Entering 134 actually produces 134.5 as a baud rate.

<bits> is a single ASCII character, which sets the number of bits per character (not counting parity). <bits> must be either 5, 6, 7, or 8. Note that the ASCII equivalent of these characters is used, so '5' is actually 35 hex.

<par> is a single ASCII character, either E, O (alphabetic "Oh"), or N, to select even, odd, or no parity.

<stop> is a single ASCII character, which selects the number of stop bits. <stop> can be either 1, 2, or 3. (Use the character '3' to select 1.5 stop bits.)

If any of the parameters is not within legal range, the entire command is ignored. However, it is possible to "fool" the firmware by entering an invalid sequence for <bits><par><stop>. An invalid sequence, but one which would not be detected as invalid, would be one in which characters from one set are interchanged with characters from another set. For example, to set 7 bits, even parity, one stop bit, <bits><par><stop> would be '7E1'. The incorrect sequence '7EE' would not be thrown out as illegal, yet would produce anomalous results. This type of incorrect sequence is not rejected, due to the way in which these characters are parsed. The moral is, "don't do this."

A sample character string would be:

```
0,1,1200,8N2
```

which would set port 0 to software handshaking, 1200 baud, 8 bits, no parity, 2 stop bits.

Opcode 5: Jump to Monitor

Offset in TRAM	Contents
0	5

Returns:

---does not return---

Opcode 5 causes entry into the SPC onboard Monitor. The Monitor uses an entirely different protocol for communication with the 68000. It is discussed in a later section of this document.

Opcode 6: Test a Port

Offset in TRAM	Contents
0	6
1	test subcode

Returns:

---depends on subcode---

Opcode 6 enters the diagnostic routines used by Chromatics' Field Service and Production departments. The diagnostic tests are designed to be used with a dedicated program (SPCTEST.SYS) running on the 68000. These diagnostics allow the serviceman to test most of the hardware on the SPC board. Tests include:

Real-time clock (NMI).

Flag 2, polled and interrupt-driven.

Flag 3, interrupt-driven.

PROM checksums.

Memory tests, both onboard and TRAM, using unique address tests, walking ones, and walking zeroes.

Port tests, including data transmit/receive at all baud rates, break send and detect, DTR and RTS outputs, DSR and DCD inputs.

Opcode 7: Load ENABLE Byte

Offset in TRAM	Contents
0	7
1	ENABLE value

Returns:

0	0

Opcode 7 loads the ENABLE cell with a 4-bit value. ENABLE is used to select which of the 4 ports is active, and is defaulted to value 0F hex. Bit 0 of this byte enables port 0, and so on.

The ENABLE cell can be altered for several reasons. Eliminating one or more ports from the processing loop will increase the time available for servicing other ports, increasing SPC throughput to some degree. This may be useful for applications in which only one or two ports are in use.

When developing programs on the SPC, and loading these programs into RAM (see Opcodes 9 through 11), it is possible that received characters could be loaded into onboard RAM and demolish your program. Setting the ENABLE byte to zero will prevent any port from being serviced, and no received characters will be buffered.

Opcode 8: Return Firmware Version

Offset in TRAM	Contents
0	8

Returns:

0	8
1	version number

Programs can use Opcode 8 to determine the revision level of SPC firmware. Versions 2 and higher support Opcode 8. (Version 1 will zero out the opcode and not provide a version number.)

This function is used primarily by the Idris operating system. Version 2 of SPC firmware, or higher, is needed to be compatible with Idris.

Opcode 9: Load Onboard Memory from TRAM

Offset in TRAM	Contents
0	9
1	destination low
2	destination high
3	count low
4	count high
5...	bytes to be loaded

Returns:

0	0

Opcode 9 downloads data from the 68000 into SPC onboard memory. This function is designed for code development. The opcode is followed in TRAM by the least significant byte of the onboard memory address, the most significant byte of the address, the count LS byte, and the count MS byte. (The Z80 uses byte-swapped notation for 16-bit numbers.)

This is followed by the bytes to be loaded into memory. No limit-checking is performed: the user must insure that all parameters are valid. It is entirely possible to bomb the SPC by loading into unsuspecting areas of RAM, and the user must know what he/she is doing.

Opcode 10: Readout onboard memory into TRAM

Offset in TRAM	Contents
0	10
1	source low
2	source high
3	count low
4	count high

Returns:

0	10
1	source low
2	source high
3	count low
4	count high
5...	bytes from onboard RAM

This command is the complement of Opcode 9. Again, no limit-checking is performed.

Opcode 11: Jump to Address

Offset in TRAM	Contents
0	11
1	address low
2	address high

Returns:

---does not return---

The use (and risk) of this command is obvious.

Opcode 12: Execute Idris Routine

Offset in TRAM	Contents
0	12

Returns:

---does not return---

Opcode 12 runs the SPC code which communicates with the Idris operating system. Idris thinks of the SPC as four devices, /dev/port0, /dev/port1, /dev/port2, and /dev/port3. See the Idris documentation for details on using these devices.

Idris divides up TRAM into four 256-byte areas, assigning one area to each port. While any port is open, the corresponding bit in ENABLE is set, allowing service for that port. The SPC periodically interrupts Idris and provides the status of each port, along with any received characters. Idris interprets this data and returns commands and characters to the SPC.

Idris can send up to 240 characters per port to the SPC in one transaction. The SPC can also send up to 240 characters to Idris. Buffering and unbuffering these characters can take a fairly long time; and during this time, the SPC might miss incoming characters. Remember that at 9600 baud, a character can arrive about once per millisecond, and if four ports are open, characters are arriving four times a millisecond. To increase receiver throughput, the Idris handler uses a coroutine to process commands from the operating system. The coroutine begins execution when WAKTIM counts to zero, and periodically pauses to allow the main Idris loop to check for received characters.

ONBOARD MONITOR

The SPC firmware contains a Z80 Monitor program, which was used during SPC software development. This Monitor program is accessable as an aid in developing user-written SPC code.

The SPC Monitor is very similar to the "CPUOS" program available in Chromatics' CG Series of color graphic computers. Current SPC firmware was developed on the CG, using Chromatics' Z80 Assembler and Text Editor. The object code was then downloaded over one of the SPC's serial ports for testing.

The following is a list of the Monitor's commands. (Refer to the CG Series manuals for detailed information.) Commands are entered as single capital letters. No delimiter is entered between the command and its first argument. A delimiter must exist between the first argument and subsequent arguments. <add1> and <add2> are hex addresses, up to 4 digits. <val> is a hex value, up to 2 digits. A delimiter must follow the complete command. Valid delimiters are the space, comma, and carriage return, except that the carriage return must not be used between arguments.

The E, H, N and P commands were contained in version 1 firmware, but were removed in version 2 to save PROM space. They were not especially useful in the SPC environment.

D - dump memory
usage: D<add1> <add2>
"D" dumps memory in hexadecimal and ASCII. If <add2> is missing, or less than <add1>, only 16 bytes are displayed.

F - fill memory
usage: F<add1> <add2> <val>
"F" fills memory from <add1> to <add2> with <val>.

G - go with breakpoints
usage: G<add1> <add2> <add3>
"G" begins execution, with optional breakpoints. Breakpoints are set at <add2> and <add3>, if they are present. Execution begins at <add1> unless it is absent, in which case execution begins at the current "PC" value (see the "X" command). If a breakpoint is hit, registers are displayed and the Monitor takes over.

K - compare memory
usage: K<add1> <add2>
"K" compares two area of memory. Any bytes which differ are displayed. After each byte, press RETURN to quit the "K" function, or any other key to proceed.

L - load object records
usage: L<add1>
"L" loads object records, in Intel hex format, into memory from serial port 0. The "L" function continues until an EOF record is found (see "E"). <add1> is an optional hex offset for the load function.

M - move data
usage: M<add1> <add2> <add3>
"M" moves bytes from the area <add1> to <add2>, to the area beginning at <add3>. After "M", "K" can be used to verify the data.

Q - search for byte
usage: Q<add1> <add2> <val> <mask>
"Q" searches the range of memory from <add1> to <add2>, for the byte <val>. Before comparing, each memory byte is masked with <mask>. This allows "Q" to search for a byte with "don't care" bits.

S - set memory
usage: S<add1>
"S" displays each byte of memory, beginning with the byte at <add1>. You may press the space key to skip that byte, or enter a new value and press the space key. Pressing the RETURN key instead of space will quit the "S" function.

X - examine registers
usage: X<reg>
"X" allows you to display and change the register values which will be used when the "G" command is given. "X" followed by RETURN displays all registers. "X" followed by a register name will display the register and allow you to enter a new value.

The following program can be used to communicate with the SPC onboard Monitor. This program runs under CGC 7900 DOS, and can be assembled on the CGC's MC68000 Resident Assembler. The program first resets the SPC, then waits for TRAM access which indicates the Z80 is running. It places opcode 5 in the first byte of TRAM, which is a command for the SPC to jump to its Monitor.

From this point on, the command protocol has changed: the Z80 becomes the host system, and the 68000 is now the terminal. The Monitor will send TRAM to the 68000 with one of two opcodes: 1 for read-character, or 2 for write-character. Our main loop processes both these opcodes by calling TERMEM's character i/o routines.

Since we call CTRLIN for character-in, we can escape from this program through a User code sequence such as the DOS or MONITOR key.

```
*
*  Program to talk to the SPC Monitor.
*
CHAROUT EQU     $800008         ;TERMEM char-out
CTRLIN  EQU     $800014         ;and char-in w/esc

FLAG1   EQU     $FF0000         ;TRAM access flag
TRAM    EQU     $FF0001         ;odd bytes only
FLAG4   EQU     $FF0004         ;reset to Z80

        ORG.L   $1C3C           ;run in DOS area

Start   CLR.B   FLAG4           ;reset the SPC
        TST.B   FLAG4           ;and let it run

        BSR     Wait            ;wait for TRAM
        MOVE.B  #5,TRAM         ;put "jmp monitor" code
        CLR.B   FLAG1           ;send it
*
*  Main loop reads/writes characters from the Monitor.
*
Main    BSR     Wait            ;wait for Monitor...
        CMP.B   #1,TRAM         ;charin request?
        BEQ.S   In
        CMP.B   #2,TRAM         ;charout request?
        BEQ.S   Out
;invalid opcode!  write an error-handler someday.
        STOP    #$2700          ;but for now, die.

In      CLR.L   D1              ;use device 0
        JSR     CTRLIN          ;(keyboard)
        BEQ.S   In              ;wait until ready
        MOVE.B  D0,TRAM+2       ;put char for Z80
        CLR.B   FLAG1           ;send it
        BRA     Main

Out     MOVE.B  TRAM+2,D0       ;get outgoing char
        CLR.L   D1              ;use device 0
        JSR     CHAROUT         ;(screen)
        CLR.B   FLAG1           ;release TRAM
        BRA     Main

Wait    BTST    #7,FLAG1        ;wait for TRAM
        BEQ.S   Wait
        RTS

        END     Start
```

DOWNLOADING CODE

Addresses 7880 through 7DFF are currently available for user-written code. Two methods are available for downloading code into the SPC, and both have been successfully used by Chromatics in developing the current firmware.

The first method uses the SPC onboard monitor. The procedure is:

Reset the SPC, to clear any previous operations.

Establish contact with the onboard monitor, using a program such as the one listed in the "Onboard Monitor" chapter of this document.

Give the "L" command, which loads object code from port 0. Port 0 is normally initialized to 9600 baud.

Transmit Intel-format hex records from a Z80 development system, such as a Chromatics CG series computer. End the data with an "end record" mark.

If all is well, the monitor prompt will return after the end record mark is detected. You may then set breakpoints and execute the downloaded code.

The second method uses opcode 9 (load onboard memory from TRAM). This method is useful when the Z80 code has been developed on the 7900, and can be downloaded through the two-port RAM. The procedure is:

Reset the SPC, to clear any previous operations.

Use opcode 7 to set the ENABLE byte to zero. This step is only necessary if you are loading into low memory (below address 7880). If so, you should insure that received characters do not get buffered on top of your program code. The way to do this is to prevent the receivers from being serviced, by zapping the ENABLE byte.

Use opcode 9 to copy your code into onboard memory. You can load up to 1019 bytes at a time.

Use opcode 11 to execute the downloaded code.

EXTERNAL CLOCKING

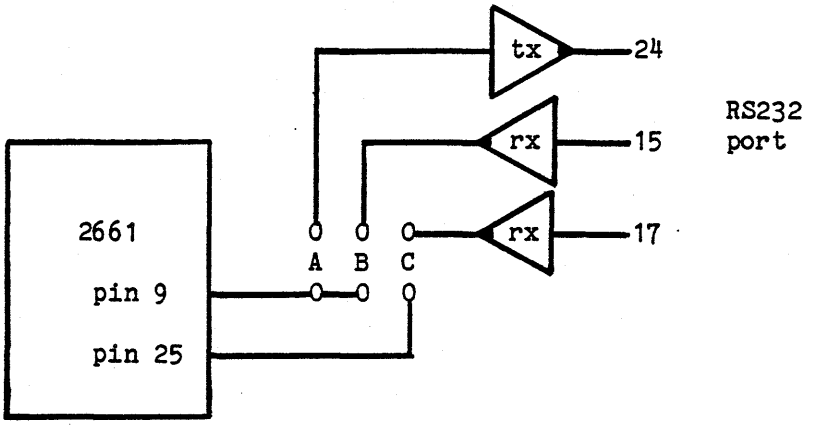
The SPC card contains jumpers which allow external clocks to feed the 2661 USARTs. The 2661's internal clocks may also be fed out to an external device. This might be necessary for a synchronous modem, for example.

> **EXTERNAL CLOCKING IS SUPPORTED BY SPC HARDWARE, BUT NOT BY SPC FIRMWARE. USE OF EXTERNAL CLOCKING WILL REQUIRE CUSTOM USER-WRITTEN FIRMWARE.** Chromatics does not provide or support firmware for external clocking (synchronous) SPC operation.

Pin 9 of the 2661 can act as a transmitter clock input, or an output from the internal baud rate generator (1x or 16x clock rate). Pin 25 of the 2661 can act as a receiver clock input, a transmitter/receiver common clock input, or a 1x or 16x clock output. Jumpers near each 2661 can be used to connect pin 9 to the RS232 port pin 15 (through a line receiver for input), or pin 15 (through a line driver for output). A jumper can also connect pin 25 of the 2661 to pin 17 of the RS232 port, through a line receiver.

> **NOTE: Custom firmware is NECESSARY before installing the jumpers.** Standard firmware will program pins 9 and 25 to be <u>output</u> pins. If the jumpers are installed, signals from the 2661 can conflict with signals from the RS232 receivers. Pins 9 and 25 are protected internally on the 2661 in case a conflict occurs, but good engineering practice will not allow the problem to arise in the first place. See the Signetics literature attached to this Application Note, and the descriptions of the 2661 contained in this Note.

The jumper configuration is as follows:



"tx" represents an RS232 transmitter, and "rx" is an RS232 receiver. 24, 15, and 17 are the RS232 connector pin numbers. A, B and C are the locations where jumpers may be installed. This configuration is repeated for each of the four SPC ports, so each port can be jumpered differently.

Pins 24, 15 and 17 are defined to be clock signals in many RS232 interfaces. Often, the arrangement is as follows:

Pin	Usage	Direction
15	Tx Clock	From Modem
17	Rx Clock	From Modem
24	Tx Clock	To Modem

If the SPC's internal clock is to be fed to external devices, install jumper A and the clock will appear on pin 24 of the RS232 connector.

For applications where external clocks must be used for both the transmitter and receiver, install jumpers B and C, and feed the clocks to the SPC on pins 15 and 17 of the RS232 connector, respectively.

If a single external clock is to be used for both transmitting and receiving, install jumper C only, and provide the external clock at pin 17.

External clock timing requirements are listed in the attached Signetics literature. Note that the RS232 transmitters and receivers perform a logical inversion of the clock signal.

DIP SWITCHES

Three 8-position DIP switches are used on the SPC to select the board's interrupt vector, interrupt priority level, base address, and card number (if more than one card is installed).

Switch SW1 selects the vector number. Position 1 on SW1 is the most significant bit of the vector number, and position 8 is the least significant bit. The low two bits of SW1 (positions 7 and 8) also select the base address of the card, either FF0000, FF0800, FF1000, or FF1800. The value set by SW1 sets the interrupt vector number, which is multiplied by 4 to determine the vector address. For example: the recommended setting for SW1 is 01111100 (zero is selected when the switch position is ON). In this case, the low two bits are 00, which set the board address at its lowest value, FF0000. The vector number is the switch value, 01111100, or 7C hex. The vector address is 7C times 4, or 1F0 hex.

Switch SW2 selects the card number in a system using multiple SPCs, in a daisy-chained interrupt configuration. (This is discussed in full in the next section.) SW2 positions 1 and 2 should be closed for the first (or only) card in a system; positions 3 and 4 must be closed for the second card, and so on. Two adjacent switches will always be closed on SW2.

Switch SW3 selects the interrupt priority level for the card. Positions 1 and 2 must be closed for level 1; positions 3 and 4 for level 2; positions 5 and 6 for level 3; and positions 7 and 8 for level 6. **Level 1 is recommended for SPC interrupts.** This is the lowest priority level available. Since the SPC performs onboard buffering, its need for service will be less than most other devices; this is why we recommend level 1. In any case, all SPC cards in a system should be at the same interrupt level, and this level must not be used by any other hardware in the system.

RECOMMENDED SWITCH SETTINGS

("X" means the switch is ON, "." means OFF.)

	Switch 1 12345678	Switch 2 12345678	Switch 3 12345678	Base Address	Vector Address
Board 0	X.....XX	XX.....	XX.....	FF0000	1F0
Board 1	X.....X.	..XX....	XX.....	FF0800	1F4
Board 2	X......X	....XX..	XX.....	FF1000	1F8
Board 3	X.......	XX	XX.....	FF1800	1FC

NOTES:

If only one board is installed in a system, it must be configured as "Board 0".

This table assumes interrupt priority level 1 is used by all SPC boards (set by SW3), and is not used by any other system hardware.

Read the next section before attempting to use multiple SPC boards in a system!

INSTALLING MULTIPLE SPCS

The SPC hardware design supports up to four cards in a system. This provides up to 16 serial ports, with each set of four ports controlled by its own Z80 processor. Note that **current 7900 firmware and SPC firmware does not support more than one card** in a system. You will have to write your own firmware to support a multiple-SPC arrangement.

All four cards should be set to the same interrupt priority level, selected by switch SW3 (see the preceding section). Alternatively, each card could be set to a different level, but this is wasteful of system resources and allows no interrupt levels for other expansion hardware. We strongly advise against this.

Assuming all four cards are at the same priority level, a mechanism is needed to arbitrate between cards when more than one card has an interrupt request pending. This mechanism is provided by SW2 and the daisy-chain connector, P7.

Switch SW2 selects a card's priority within the daisy-chain. Board 0 (see the preceding section) will have the highest priority of the group, followed by board 1 and 2. Board 3 will have the lowest priority of the group. SW2 must be set consecutively for each of the boards in a system.

The daisy-chain cable is connected to P7, and runs in parallel to all boards. This cable is constructed of 10-conductor ribbon cable, and 10-position card edge connectors. It should be as short as possible for best noise rejection. Ideally, all SPCs in a system will be located in adjacent card slots.

The daisy-chain mechanism works as follows: when any card has an interrupt request pending, all cards below it in the chain are prevented from responding to interrupt acknowledge. The next INTACK (interrupt acknowledge) from the CPU, of the correct priority level, will be responded to by the highest SPC in the chain with an interrupt request pending. After the INTACK is complete, other cards in the chain are again enabled. SW2 and P7 provide a gated path for INTACK between cards.

As a result, the highest board in a chain (board 0) will, ideally, get slightly more attention from the 68000 than the other boards. If this is a concern, you should connect the most important devices to the ports on board 0, and the least important devices to board 3.

Software interrupt handlers, running in the 68000, can use much of the same code for all four SPC boards. The interrupt vector (1F0 to 1FC) will determine the base address of the interrupting board (FF0000 to FF1800). Since all four cards are at the same interrupt priority level, there is no concern that the routine will be re-interrupted by another SPC interrupt request. Re-entrant code should not be required.

PORT PINOUT

Each of the four SPC ports uses a male 25-pin "D" connector. The port is wired as a terminal.

To connect to a modem, use a straight cable, wired one-to-one, with a female connector on the SPC end and a male on the modem end. To connect to a terminal, you must construct a cable which interchanges pins 2 and 3, pins 4 and 5, and pins 6 and 20.

In some applications, only pins 2, 3, and 7 are necessary for proper operation.

Pin #	Usage
1	Ground
2	Transmit Data (output)
3	Receive Data (input)
4	Request to Send (output)
5	Clear To Send (input)
6	Data Set Ready (output)
7	Ground
8	Data Carrier Detect (input)
20	Data Terminal Ready (output)
15, 17, 24	User-defined (see below)

Pins 5, 6, and 8 are control inputs. They are normally driven by the corresponding outputs of a modem. Internal pullup resistors on the SPC will hold these signals in a "true" state if the external device does not connect to them.

Pin 4 (RTS) is always asserted "true" by standard firmware, when the SPC is running. Pin 20 (DTR) is also "true" unless hardware handshaking is in use; then it becomes "false" when the SPC is unable to accept characters. Idris also uses pin 20 as a modem-control signal.

Pins 15, 17 and 24 are disconnected unless jumpers are installed. See the "External Clocking" section of this document for details.

Other pins are not connected.

2661 DATA SHEET

The following material is reprinted by permission of Signetics Corporation, a subsidiary of U.S. Philips Corporation, 1077 East Arques Avenue, Sunnyvale California 94086.

Copyright (c) 1981 by Signetics Corporation.

Reprinted by Chromatics, Inc. with Permission of Signetics Corp.,
a subsidiary of U.S. Philips Corp., 1077 E. Arques Ave.,
Sunnyvale, Ca 94086.

Signetics

SC2661
November 1981

Enhanced Programmable Communications Interface (EPCI)

Signetics reserves the right to make changes in the products contained in this document in order to improve design or performance and to supply the best possible products. Signetics also assumes no responsibility for the use of any circuits described herein, conveys no license under any patent or other right, and makes no representations that the circuits are free from patent infringement. Applications for any integrated circuits contained in this publication are for illustration purposes only and Signetics makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification. Reproduction of any portion hereof without the prior written consent of Signetics is prohibited.

ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE (EPCI) SC2661

DESCRIPTION

The Signetics 2661 EPCI is a universal synchronous/asynchronous data communications controller chip that is an enhanced pin compatible version of the 2651. It interfaces directly to most 8-bit microprocessors and may be used in a polled or interrupt driven system environment. The 2661 accepts programmed instructions from the microprocessor while supporting many serial data communications disciplines—synchronous and asynchronous—in the full or half-duplex mode. Special support for BISYNC is provided.

The EPCI serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The 2661 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode. Each version of the EPCI (A, B, C) has a different set of baud rates.

The EPCI is constructed using Signetics n-channel silicon gate depletion load technology and is packaged in a 28-pin DIP.

FEATURES

- **Synchronous operation**
  - **5 to 8-bit characters plus parity**
  - **Single or double SYN operation**
  - **Internal or external character synchronization**
  - **Transparent or non-transparent mode**
  - **Transparent mode DLE stuffing (Tx) and detection (Rx)**
  - **Automatic SYN or DLE-SYN insertion**
  - **SYN, DLE and DLE-SYN stripping**
  - **Odd, even, or no parity**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
- **Asynchronous operation**
  - **5 to 8-bit characters plus parity**
  - **1, 1½ or 2 stop bits transmitted**
  - **Odd, even, or no parity**
  - **Parity, overrun and framing error detection**
  - **Line break detection and generation**
  - **False start bit detection**
  - **Automatic serial echo mode (echoplex)**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
    dc to 62.5K bps (16X clock)
    dc to 15.625K bps (64X clock)

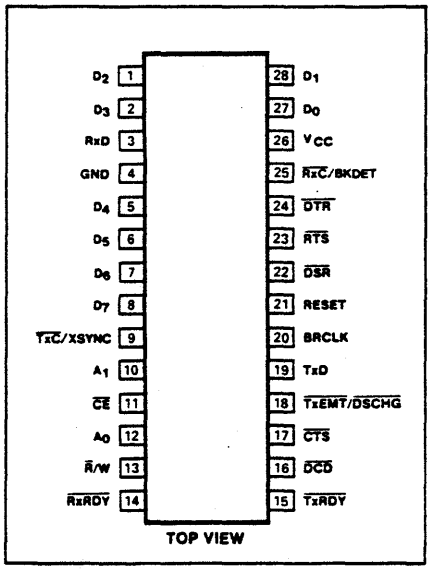
OTHER FEATURES

- **Internal or external baud rate clock**
- **3 baud rate sets**
- **16 internal rates for each set**
- **Double buffered transmitter and receiver**
- **Dynamic character length switching**
- **Full or half duplex operation**
- **Fully compatible with 2650 CPU**
- **TTL compatible inputs and outputs**
- **RxC and TxC pins are short circuit protected**
- **3 open drain MOS outputs can be wire-ORed**
- **Single 5V power supply**
- **No system clock required**
- **28-pin dual in-line package**

APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front end processors**
- **Remote data concentrators**
- **Computer to computer links**
- **Serial peripherals**
- **BISYNC adaptors**

PIN CONFIGURATION



ORDERING CODE

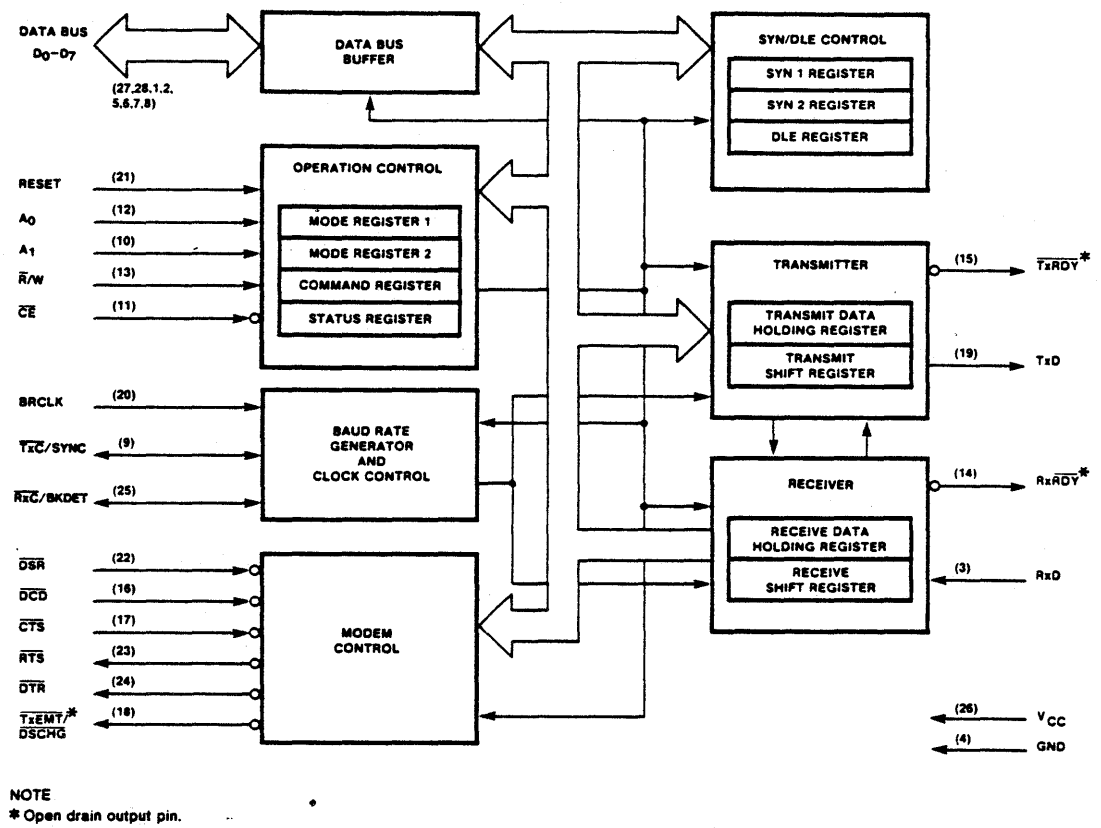
PACKAGES	COMMERCIAL RANGES V_{CC} = 5V ±5%, T_A = 0°C to 70°C	
Ceramic DIP	SC2661ACSI28 SC2661BCSI28 SC2661CCSI28	See table 1 for baud rates
Plastic DIP	SC2661ACSN28 SC2661BCSN28 SC2661CCSN28	See table 1 for baud rates

PIN DESIGNATION

PIN NO.	SYMBOL	NAME AND FUNCTION	TYPE
27,28,1, 2,5-8	D_0-D_7	8-bit data bus	I/O
21	RESET	Reset	I
12,10	A_0-A_1	Internal register select lines	I
13	$\overline{R}/W$	Read or write command	I
11	$\overline{CE}$	Chip enable input	I
22	$\overline{DSR}$	Data set ready	I
24	$\overline{DTR}$	Data terminal ready	O
23	$\overline{RTS}$	Request to send	O
17	$\overline{CTS}$	Clear to send	I
16	$\overline{DCD}$	Data carrier detected	I
18	$\overline{TxEMT}/\overline{DSCHG}$	Transmitter empty or data set change	O
9	$\overline{TxC}$/XSYNC	Transmitter clock/external SYNC	I/O
25	$\overline{RxC}$/BKDET	Receiver clock/break detect	I/O
19	TxD	Transmitter data	O
3	RxD	Receiver data	I
15	$\overline{TxRDY}$	Transmitter ready	O
14	$\overline{RxRDY}$	Receiver ready	O
20	BRCLK	Baud rate generator clock	I
26	V_{CC}	+5V supply	I
4	GND	Ground	I

ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE (EPCI) SC2661

BLOCK DIAGRAM



NOTE
* Open drain output pin.

BLOCK DIAGRAM

The EPCI consists of six major sections. These are the transmitter, receiver, timing, operation control, modem control and SYN/DLE control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing and protocol are presented in the EPCI programming section of this data sheet.

Table 1 BAUD RATE GENERATOR CHARACTERISTICS SC2661A (BRCLK = 4.9152MHz)

MR23-20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	-	6144
0001	75	1.2	-	4096
0010	110	1.7598	−0.01	2793
0011	134.5	2.152	-	2284
0100	150	2.4	-	2048
0101	200	3.2	-	1536
0110	300	4.8	-	1024
0111	600	9.6	-	512
1000	1050	16.8329	0.196	292
1001	1200	19.2	-	256
1010	1800	28.7438	−0.19	171
1011	2000	31.9168	−0.26	154
1100	2400	38.4	-	128
1101	4800	76.8	-	64
1110	9600	153.6	-	32
1111	19200	307.2	-	16

Timing

The EPCI contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full duplex operation. See table 1.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU.

Transmitter

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for three input signals and three output signals used for "handshaking" and status indication between the CPU and a modem.

SYN/DLE Control

This section contains control circuitry and three 8-bit registers storing the SYN1, SYN2, and DLE characters provided by the CPU. These registers are used in the synchronous mode of operation to provide the characters required for synchronization, idle fill and data transparency.

Table 1 BAUD RATE GENERATOR CHRACTERISTICS (Cont'd)

SC2661B (BRCLK = 4.9152MHz)

MR23-20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	45.5	0.7279kHz	0.005	6752
0001	50	0.8	-	6144
0010	75	1.2	-	4096
0011	110	1.7598	−0.01	2793
0100	134.5	2.152	-	2284
0101	150	2.4	-	2048
0110	300	4.8	-	1024
0111	600	9.6	-	512
1000	1200	19.2	-	256
1001	1800	28.7438	−0.19	171
1010	2000	31.9168	−0.26	154
1011	2400	38.4	-	128
1100	4800	76.8	-	64
1101	9600	153.6	-	32
1110	19200	307.2	-	16
1111	38400	614.4	-	8

SC2661C (BRCLK = 5.0688MHz)

MR23-20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	-	6336
0001	75	1.2	-	4224
0010	110	1.76	-	2880
0011	134.5	2.1523	0.016	2355
0100	150	2.4	-	2112
0101	300	4.8	-	1056
0110	600	9.6	-	528
0111	1200	19.2	-	264
1000	1800	28.8	-	176
1001	2000	32.081	0.253	158
1010	2400	38.4	-	132
1011	3600	57.6	-	88
1100	4800	76.8	-	66
1101	7200	115.2	-	44
1110	9600	153.6	-	33
1111	19200	316.8	3.125	16

NOTE
16X clock is used in asynchronous mode. In synchronous mode, clock multiplier is 1X and BRG can be used only for TxC.

Table 2 CPU-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/ OUTPUT	FUNCTION
V_{CC}	26	I	+5V supply input
GND	4	I	Ground
RESET	21	I	A high on this input performs a master reset on the 2661. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with the appropriate control words.
A_1-A_0	10,12	I	Address lines used to select internal EPCI registers.
$\overline{R}/W$	13	I	Read command when low, write command when high.
$\overline{CE}$	11	I	Chip enable command. When low, indicates that control and data lines to the EPCI are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the D_0-D_7 lines in the three-state condition.
D_7-D_0	8,7,6,5, 2,1,28,17	I/O	8-bit, three-state data bus used to transfer commands, data and status between EPCI and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{TxRDY}$	15	O	This output is the complement of status register bit SR0. When low, it indicates that the transmit data holding register (THR) is ready to accept a data character from the CPU. It goes high when the data character is loaded. This output is valid only when the transmitter is enabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{RxRDY}$	14	O	This output is the complement of status register bit SR1. When low, it indicates that the receive data holding register (RHR) has a character ready for input to the CPU. It goes high when the RHR is read by the CPU, and also when the receiver is disabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{TxEMT}$/ $\overline{DSCHG}$	18	O	This output is the complement of status register bit SR2. When low, it indicates that the transmitter has completed serialization of the last character loaded by the CPU, or that a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs has occurred. This output goes high when the status register is read by the CPU, if the TxEMT condition does not exist. Otherwise, the THR must be loaded by the CPU for this line to go high. It is an open drain output which can be used as an interrupt to the CPU.

OPERATION

The functional operation of the 2661 is programmed by a set of control words supplied by the CPU. These control words specify items such as synchronous or asynchronous mode, baud rate, number of bits per character, etc. The programming procedure is described in the EPCI programming section of the data sheet.

After programming, the EPCI is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The 2661 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. In the asynchronous mode, the receiver looks for a high to low (mark to space) transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and one stop bit have been assembled. The data are then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{RxRDY}$ output is asserted. If the character length is less than 8 bits, the high order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive going edge of $\overline{RxC}$ corresponding to the received character boundary. If the stop bit is present, the receiver will immediately begin its search for the next start bit. If the stop bit is absent (framing error), the receiver will interpret a space as a start bit if it persists into the next bit time interval. If a break condition is detected (RxD is low for the entire character as well as the stop bit), only one character consisting of all zeros (with the FE status bit SR5 set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

Pin 25 can be programmed to be a break detect output by appropriate setting of MR27-MR24. If so, a detected break will cause that pin to go high. When RxD returns to mark for one RxC time, pin 25 will go low. Refer to the break detection timing diagram.

Table 3 DEVICE-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/ OUTPUT	FUNCTION
BRCLK	20	I	Clock input to the internal baud rate generator (see table 1). Not required if external receiver and transmitter clocks are used.
*$\overline{RxC}$/BKDET	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data are sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin can be a 1X/16X clock or a break detect output pin.
*$\overline{TxC}$/XSYNC	9	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin can be a 1X/16X clock output or an external jam synchronization input.
RxD	3	I	Serial data input to the receiver. "Mark" is high, "space" is low.
TxD	19	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{DSR}$	22	I	General purpose input which can be used for data set ready or ring indicator condition. Its complement appears as status register bit SR7. Causes a low output on $\overline{TxEMT}$/$\overline{DSCHG}$ when its state changes if CR2 or CR0 = 1.
$\overline{DCD}$	16	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{TxEMT}$/$\overline{DSCHG}$ when its state changes if CR2 or CR0 = 1. If $\overline{DCD}$ goes high while receiving, the RxC is internally inhibited.
$\overline{CTS}$	17	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{DTR}$	24	O	General purpose output which is the complement of command register bit CR1. Normally used to indicate data terminal ready.
$\overline{RTS}$	23	O	General purpose output which is the complement of command register bit CR5. Normally used to indicate request to send. If the transmit shift register is not empty when CR5 is reset (1 to 0), then $\overline{RTS}$ will go high one TxC time after the last serial bit is transmitted.

NOTE
*$\overline{RxC}$ and $\overline{TxC}$ outputs have short circuit protection max. C_L = 100pF. Outputs become open circuited upon detection of a zero pulled high or a one pulled low.

When the EPCI is initialized into the synchronous mode, the receiver first enters the hunt mode on a 0 to 1 transition of RxEN(CR2). In this mode, as data are shifted into the receiver shift register a bit at a time, the contents of the register are compared to the contents of the SYN1 register. If the two are not equal, the next bit is shifted in and the comparison is repeated. When the two registers match, the hunt mode is terminated and character assembly mode begins. If single SYN operation is programmed, the SYN DETECT status bit is set. If double SYN operation is programmed, the first character assembled after SYN1 must be SYN2 in order for the SYN DETECT bit to be set. Otherwise, the EPCI returns to the hunt mode. (Note that the sequence SYN1-SYN1-SYN2 will not achieve synchronization.) When synchronization has been achieved, the EPCI continues to assemble characters and transfer them to the holding register, setting the RxRDY status bit and asserting the $\overline{RxRDY}$ output each time a character is transferred. The PE and OE status bits are set as appropriate. Further receipt of the appropriate SYN sequence sets the SYN DETECT status bit. If the SYN stripping mode is commanded, SYN characters are not transferred to the holding register. Note that the SYN characters used to establish initial synchronization are not transferred to the holding register in any case.

External jam synchronization can be achieved via pin 9 by appropriate setting of MR27-MR24. When pin 9 is an XSYNC input, the internal SYN1, SYN1-SYN2, and DLE-SYN1 detection is disabled. Each positive going signal on XSYNC will cause the receiver to establish synchronization on the rising edge of the next RxC pulse. Character assembly will start with the RxD input at this edge. XSYNC may be lowered on the next rising edge of RxC. This external synchronization will cause the SYN DETECT status bit to be set until the status register is read. Refer to XSYNC timing diagram.

Transmitter

The EPCI is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The 2661 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{TxRDY}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data are transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

In the asynchronous mode, the transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the $\overline{TxEMT}/\overline{DSCHG}$ output and its corresponding status bit are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit (CR3) high.

In the synchronous mode, when the 2661 is initially conditioned to transmit, the TxD output remains high and the TxRDY condition is asserted until the first character to be transmitted (usually a SYN character) is loaded by the CPU. Subsequent to this, a continuous stream of characters is transmitted. No extra bits (other than parity, if commanded) are generated by the EPCI unless the CPU fails to send a new character to the EPCI by the time the transmitter has completed sending the previous character. Since synchronous communication does not allow gaps between characters, the EPCI asserts TxEMT and automatically "fills" the gap by transmitting SYN1s, SYN1-SYN2 doublets, or DLE-SYN1 doublets, depending on the state of MR16 and MR17. Normal transmission of the message resumes when a new character is available in the transmit data holding register. If the SEND DLE bit in the command register is true, the DLE character is automatically transmitted prior to transmission of the message character in the THR.

EPCI PROGRAMMING

Prior to initiating data communications, the 2661 operational mode must be programmed by performing write operations to the mode and command registers. In addition, if synchronous operation is programmed, the appropriate SYN/DLE registers must be loaded. The EPCI can be reconfigured at any time during program execution. A flowchart of the intialization process appears in figure 1.

The internal registers of the EPCI are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

The SYN1, SYN2, and DLE registers are accessed by performing write operations with the conditions $A_1 = 0$, $A_0 = 1$, and

Table 4 2661 REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Three-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Write SYN1/SYN2/DLE registers
0	1	0	0	Read mode registers ½
0	1	0	1	Write mode registers ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE
See AC characteristics section for timing requirements.

2661 INITIALIZATION FLOW CHART

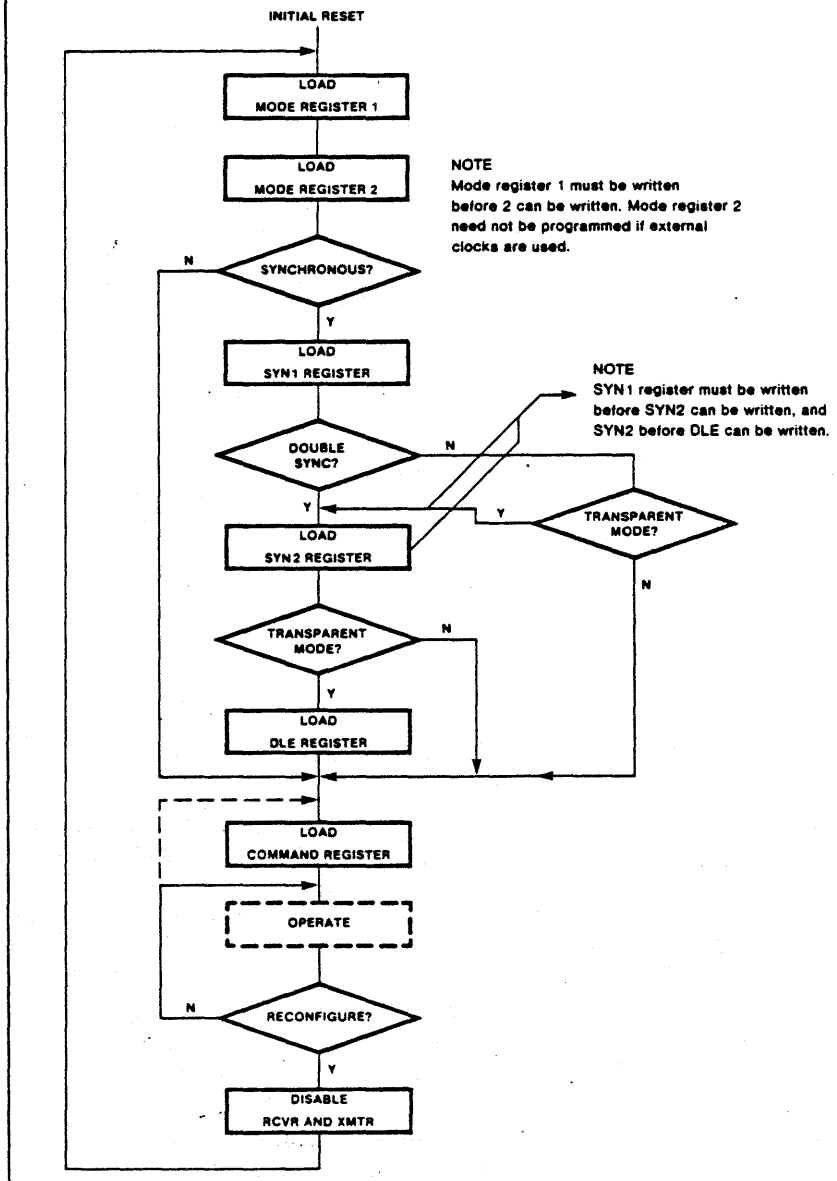


Figure 1

$\overline{R}/W$ = 1. The first operation loads the SYN1 register. The next loads the SYN2 register, and the third loads the DLE register. Reading or loading the mode registers is done in a similar manner. The first write (or read) operation addresses mode register 1, and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointers are reset to SYN1 register and mode register 1 by a RESET input or by performing a read command register operation, but are unaffected by any other read or write operation.

The 2661 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the EPCI, while the command register controls the operation within this basic framework. The EPCI indicates its status in the status register. These registers are cleared when a RESET input is applied.

Mode Register 1 (MR1)

Table 5 illustrates Mode Register 1. Bits MR11 and MR10 select the communication format and baud rate multiplier. 00 specifies synchronous mode and 1X multiplier. 1X, 16X, and 64X multipliers are programmable for asynchronous format. However, the multiplier in asynchronous format applies only if the external clock input option is selected by MR24 or MR25.

MR13 and MH12 select a character length of 5, 6, 7 or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits in asynchronous mode.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

In asynchronous mode, MR17 and MR16 select character framing of 1, 1.5, or 2 stop bits. (If 1X baud rate is programmed, 1.5 stop bits defaults to 1 stop bits on transmit.) In synchronous mode, MR17 controls the number of SYN characters used to establish synchronization and for character fill when the transmitter is idle. SYN1 alone is used if MR17 = 1, and SYN1-SYN2 is used when MR17 = 0. If the transparent mode is specified by MR16, DLE-SYN1 is used for character fill and SYN detect, but the normal synchronization sequence is used to establish character sync. When transmitting, a DLE character in the transmit holding register will cause a second DLE character to be transmitted. This DLE stuffing eliminates the software DLE compare and stuff on each transparent mode data character. If the send DLE command (CR3) is active when a DLE is loaded into THR, only one additional DLE will be transmitted. Also, DLE stripping and DLE detect (with MR14 = 0) are enabled.

The bits in the mode register affecting character assembly and disassembly (MR12-MR16) can be changed dynamically (during active receive/transmit operation). The character mode register affects both the transmitter and receiver; therefore in synchronous mode, changes should be made only in half duplex mode (RxEN = 1 or TxEN = 1, but not both simultaneously = 1). In asynchronous mode, character changes should be made when RxEN and TxEN=0 or when TxEN = 1 and the transmitter is marking in half duplex mode (RxEN = 0).

To effect assembly/disassembly of the next received/transmitted character, MR12-15 must be changed within n bit times of the active going state of $\overline{RxRDY}/\overline{TxRDY}$. Transparent and non-transparent mode changes (MR16) must occur within n-1 bit times of the character to be affected when the receiver or transmitter is active. (n = smaller of the new and old character lengths.)

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21 and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable for each EPCI version (−1, −2, −3). Version 1 and 2 specify a 4.9152 MHz TTL input at BRCLK (pin 20); version 3 specifies a 5.0688 MHz input which is identical to the Signetics 2651. MR23-20 are don't cares if external clocks are selected (MR25-MR24 = 0). The individual rates are given in table 1.

MR24-MR27 select the receive and transmit clock source (either the BRG or an external input) and the function at pins 9 and 25. Refer to table 6.

Command Register (CR)

Table 7 illustrates the command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. A 0 to 1 transition of CR2 forces start bit search (async mode) or hunt mode (sync mode) on the second $\overline{RxC}$ rising edge. Disabling the receiver causes $\overline{RxRDY}$ to go high (inactive). If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state

Table 5 MODE REGISTER 1 (MR 1)

<table>
<tr><th colspan="2">MR17 MR16</th><th>MR15</th><th>MR14</th><th>MR13 MR12</th><th>MR11 MR10</th></tr>
<tr><td colspan="2">Sync/Async</td><td>Parity Type</td><td>Parity Control</td><td>Character Length</td><td>Mode and Baud Rate Factor</td></tr>
<tr><td colspan="2">Async: Stop Bit Length
00 = Invalid
01 = 1 stop bit
10 = 1½ stop bits
11 = 2 stop bits</td><td rowspan="2">0 = Odd
1 = Even</td><td rowspan="2">0 = Disabled
1 = Enabled</td><td rowspan="2">00 = 5 bits
01 = 6 bits
10 = 7 bits
11 = 8 bits</td><td rowspan="2">00 = Synchronous 1X rate
01 = Asynchronous 1X rate
10 = Asynchronous 16X rate
11 = Asynchronous 64X rate</td></tr>
<tr><td>Sync:
Number of SYN char
0 = Double SYN
1 = Single SYN</td><td>Sync:
Transparency Control
0 = Normal
1 = Transparent</td></tr>
</table>

NOTE
Baud rate factor in asynchronous applies only if external clock is selected. Factor is 16X if internal clock is selected. Mode must be selected (MR11, MR10) in any case.

Table 6 MODE REGISTER 2 (MR2)

MR27–MR24											MR23–MR20
	TxC	RxC	Pin 9	Pin 25		TxC	RxC	Pin 9	Pin 25	Mode	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	XSYNC[1]	RxC/TxC	sync	
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	async	
0010	I	E	1X	RxC	1010	I	E	XSYNC[1]	RxC	sync	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	async	See baud rates in table 1
0100	E	E	TxC	RxC	1100	E	E	XSYNC[1]	RxC/TxC	sync	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	async	
0110	I	E	16X	RxC	1110	I	E	XSYNC[1]	RxC	sync	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	async	

NOTES
1. When pin 9 is programmed as XSYNC input, SYN1, SYN1-SYN2, and DLE-SYN1 detection is disabled.

E = External clock
I = Internal clock (BRG)
1X and 16X are clock outputs

Table 7 COMMAND REGISTER (CR)

CR7 CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode	Request To Send	Reset Error	Sync/Async	Receive Control (RxEN)	Data Terminal Ready	Transmit Control (TxEN)
00 = Normal operation 01 = Async: Automatic echo mode Sync: SYN and/or DLE stripping mode 10 = Local loop back 11 = Remote loop back	0 = Force $\overline{RTS}$ output high one clock time after TxSR serialization 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flags in status register (FE, OE, PE/DLE detect)	Async: Force break 0 = Normal 1 = Force break	0 = Disable 1 = Enable	0 = Force $\overline{DTR}$ output high 1 = Force $\overline{DTR}$ output low	0 = Disable 1 = Enable
			Sync: Send DLE 0 = Normal 1 = Send DLE			

Table 8 STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	Data Carrier Detect	FE/SYN Detect	Overrun	PE/DLE Detect	TxEMT/DSCHG	RxRDY	TxRDY
0 = $\overline{DSR}$ input is high 1 = $\overline{DSR}$ input is low	0 = $\overline{DCD}$ input is high 1 = $\overline{DCD}$ input is low	Async: 0 = Normal 1 = Framing Error	0 = Normal 1 = Overrun Error	Async: 0 = Normal 1 = Parity error	0 = Normal 1 = Change in $\overline{DSR}$, or $\overline{DCD}$, or transmit shift register is empty	0 = Receive holding register empty 1 = Receive holding register has data	0 = Transmit holding register busy 1 = Transmit holding register empty
		Sync: 0 = Normal 1 = SYN detected		Sync: 0 = Normal 1 = Parity error or DLE received			

(high) while $\overline{TxRDY}$ and $\overline{TxEMT}$ will go high (inactive). If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be neglected. A 0 to 1 transition of CR2 will initiate start bit search (async) or hunt mode (sync).

Bits CR1 (DTR) and CR5 (RTS) control the $\overline{DTR}$ and $\overline{RTS}$ outputs. Data at the outputs are the logical complement of the register data.

In asynchronous mode, setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The TxD line will go high for at least one bit time before beginning transmission of the next character in the transmit data holding register. In synchronous mode, setting CR3 causes the transmission of the DLE register contents prior to sending the character in the transmit data holding register. Since this is a one time command, CR3 does not have to be reset by software. CR3 should be set when entering and exiting transparent mode and for all DLE—non-DLE character sequences.

Setting CR4 causes the error flags in the status register (SR3, SR4, and SR5) to be cleared. This is a one time command. There is no internal latch for this bit.

Table 9 SC2661 EPCI vs SC2651 PCI

FEATURE	EPCI	PCI
1. MR2 Bit 6, 7	Control pin 9, 25	Not used
2. DLE detect-SR3	SR3 = 0 for DLE-DLE, DLE-SYNC1	SR3 = 1 for DLE-DLE, DLE-SYNC1
3. Reset of SR3, DLE detect	Second character after DLE, or receiver disable, or CR4 = 1	Receiver disable, or CR4 = 1
4. Send DLE-CR3	One time command	Reset via CR3 on next $\overline{TxRDY}$
5. DLE stuffing in transparent mode	Automatic DLE stuffing when DLE is loaded except if CR3 = 1	None
6. SYNC1 stripping in double sync non-transparent mode	All SYNC1	First SYNC1 of pair
7. Baud rate versions	Three	One
8. Terminate ASYNC transmission (drop RTS)	Reset CR5 in response to $\overline{TxRDY}$ changing from 1 to 0	Reset CR0 when $\overline{TxEMT}$ goes from 1 to 0. Then reset CR5 when $\overline{TxEMT}$ goes from 0 to 1
9. Break detect	Pin 25[1]	FE and null character
10. Stop bit searched	One	Two
11. External jam sync	Pin 9[2]	No
12. Data bus timing	Improved over 2651	—
13. Data bus drivers	Sink 2.2mA Source 400μA	Sink 1.6mA Source 100μA

NOTES
1. Internal BRG used for RxC.
2. Internal BRG used for TxC.

When CR5 (RTS) is set, the $\overline{RTS}$ pin is forced low and the transmit serial logic is enabled. A 1 to 0 transition of CR5 will cause $\overline{RTS}$ to go high (inactive) one TxC time after the last serial bit has been transmitted (if the transmit shift register was not empty).

The EPCI can operate in one of four sub-modes within each major mode (synchronous or asynchronous). The operational sub-mode is determined by CR7 and CR6. CR7-CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

In asynchronous mode, CR7-CR6 = 01 places the EPCI in the automatic echo mode. Clocked, regenerated received data are automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU to receiver communications continues normally, but the CPU to transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. $\overline{TxRDY}$ output = 1.
4. The $\overline{TxEMT}$/DSCHG pin will reflect only the data set change condition.
5. The TxEN command (CR0) is ignored.

In synchronous mode, CR7-CR6 = 01 places the EPCI in the automatic SYN/DLE stripping mode. The exact action taken depends on the setting of bits MR17 and MR16:

1. In the non-transparent, single SYN mode (MR17-MR16 = 10), characters in the data stream matching SYN1 are not transferred to the receive data holding register (RHR).
2. In the non-transparent, double SYN mode (MR17-MR16 = 00), characters in the data stream matching SYN1, or SYN2 if immediately preceded by SYN1, are not transferred to the RHR.
3. In transparent mode (MR16 = 1), characters in the data stream matching DLE, or SYN1 if immediately preceded by DLE, are not transferred to the RHR. However, only the first DLE of a DLE-DLE pair is stripped.

Note that automatic stripping mode does not affect the setting of the DLE detect and SYN detect status bits (SR3 and SR5).

Two diagnostic sub-modes can also be configured. In local loop back mode (CR7-CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{DTR}$ is connected to $\overline{DCD}$ and $\overline{RTS}$ is connected to $\overline{CTS}$.
3. The receiver is clocked by the transmit clock.
4. The $\overline{DTR}$, $\overline{RTS}$ and $\overline{TxD}$ outputs are held high.
5. The $\overline{CTS}$, $\overline{DCD}$, $\overline{DSR}$ and RxD inputs are ignored.

Additional requirements to operate in the local loop back mode are that CR0 (TxEN), CR1 (DTR), and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the EPCI.

The second diagnostic mode is the remote loop back mode (CR7-CR6 = 11). In this mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data are sent to the local CPU, but the error status conditions (PE, OE, FE) are set.
4. The $\overline{RxRDY}$, $\overline{TxRDY}$, and $\overline{TxEMT}$/$\overline{DSCHG}$ outputs are held high.
5. CR1 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It, and its corresponding output, are valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit data holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the holding register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the automatic echo or remote loopback modes are programmed. When this bit is set, the $\overline{TxRDY}$ output pin is low. In

the automatic echo and remote loop back modes, the output is held high.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to zero, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{RxRDY}$ output is low.

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs (when CR2 or CR0 = 1) or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. Note that in synchronous mode this bit will be set even though the appropriate "fill" character is transmitted. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when TxEN = 1 or RxEN = 1. It is cleared when the status register is read by the CPU. If the status register is read twice and SR2 = 1 while SR6 and SR7 remain unchanged, then a TxEMT condition exists. When SR2 is set, the $\overline{TxEMT}/\overline{DSCHG}$ output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. In synchronous transparent mode (MR16 = 1), with parity disabled, it indicates that a character matching DLE register was received and the present character is neither SYN1 nor DLE. This bit is cleared when the next character following the above sequence is loaded into RHR, when the receiver is disabled, or by a reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled or by the reset error command, CR4.

In asynchronous mode, bit SR5 signifies that the received character was not framed by a stop bit, i.e., only the first stop bit is checked. If RHR = 0 when SR5 = 1, a break condition is present. In synchronous non-transparent mode (MR16 = 0), it indicates receipt of the SYN1 character in single SYN mode or the SYN1-SYN2 pair in double SYN mode. In synchronous transparent mode (MR16 = 1), this bit is set upon detection of the initial synchronizing characters (SYN1 or SYN1-SYN2) and, after synchronization has been achieved, when a DLE-SYN1 pair is received. The bit is reset when the receiver is disabled, when the reset error command is given in asynchronous mode, or when the status register is read by the CPU in the synchronous mode.

SR6 and SR7 reflect the conditions of the $\overline{DCD}$ and $\overline{DSR}$ inputs respectively. A low input sets its corresponding status bit, and a high input clears it.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5.0V ±5% [4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Input voltage					V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
	Output voltage					V
V_{OL}	Low	I_{OL} = 2.2mA			0.4	
V_{OH}[7]	High	I_{OH} = −400μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to 5.5 V			10	μA
	3-state output leakage current					μA
I_{LH}	Data bus high	V_O = 4.0V			10	
I_{LL}	Data bus low	V_O = 0.45V			10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Capacitance					pF
C_{IN}	Input				20	
C_{OUT}	Output	fc = 1MHz			20	
$C_{I/O}$	Input/Output	Unmeasured pins tied to ground			20	

Notes on following page.

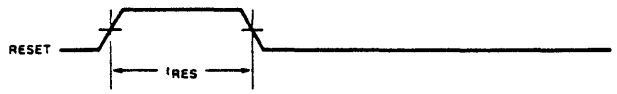
AC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5.0V ± 5% [4,5,6]

	PARAMETER	TEST CONDITIONS	Min	Typ	Max	UNIT
	Pulse width					ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		250			
	Setup and hold time					ns
t_{AS}	Address setup		10			
t_{AH}	Address hold		10			
t_{CS}	$\overline{R}$/W control setup		10			
t_{CH}	$\overline{R}$/W control hold		10			
t_{DS}	Data setup for write		150			
t_{DH}	Data hold for write		0			
t_{RXS}	Rx data setup		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	C_L = 150pF			200	ns
t_{DF}	Data bus floating time for read	C_L = 150pF			100	
t_{CED}	CE to CE delay		600			
	Input clock frequency					MHz
f_{BRG}	Baud rate generator (2661A,B)		1.0	4.9152	4.9202	
f_{BRG}	Baud rate generator (2661C)		1.0	5.0688	5.0738	
$f_{R/T}$ [10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
	Clock width					ns
t_{BRH} [9]	Baud rate high (2661A,B)		75			
t_{BRH} [9]	Baud rate high (2661C)		70			
t_{BRL} [9]	Baud rate low (2661A,B)		75			
t_{BRL} [9]	Baud rate low (2661C)		70			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		480			
$t_{R/TL}$ [10]	$\overline{TxC}$ or $\overline{RxC}$ low		480			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	C_L = 150pF			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output [8]	C_L = 150pF		0		

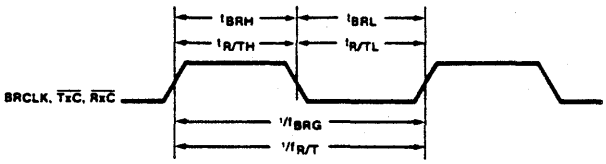
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient (IQ ceramic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and t_{BRL}) and at 0.8V and 2.0V for outputs. Input levels swing between 0.4V and 2.4V, with a transition time of 20 ns maximum.
6. Typical values are at +20°C, typical supply voltages and typical processing parameters.
7. $\overline{TxRDY}$, $\overline{RxRDY}$ and $\overline{TxEMT/DSCHG}$ outputs are open drain.
8. Parameter applies when internal transmitter clock is used.
9. Under test conditions of 5.0688 MHz f_{BRG} (2661C) and 4.9152 MHz f_{BRG} (2661A,B), t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. In asynchronous local loopback mode, using 1X clock, the following parameters apply:
    $f_{R/T}$ = 0.83 MHz max.
    $t_{R/TL}$ = 700 ns min.

TIMING DIAGRAMS

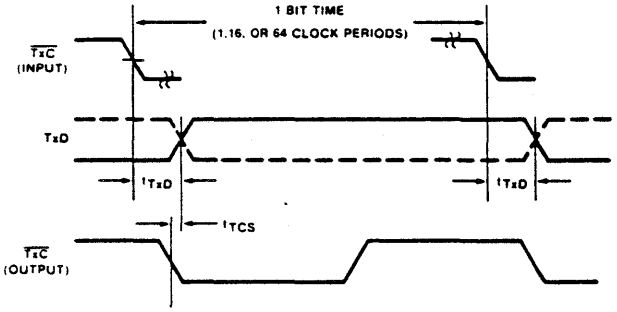
RESET



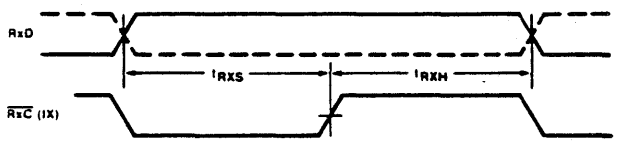
CLOCK



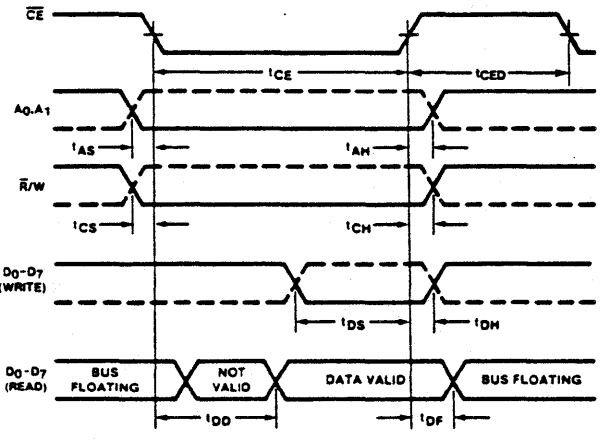
TRANSMIT



RECEIVE

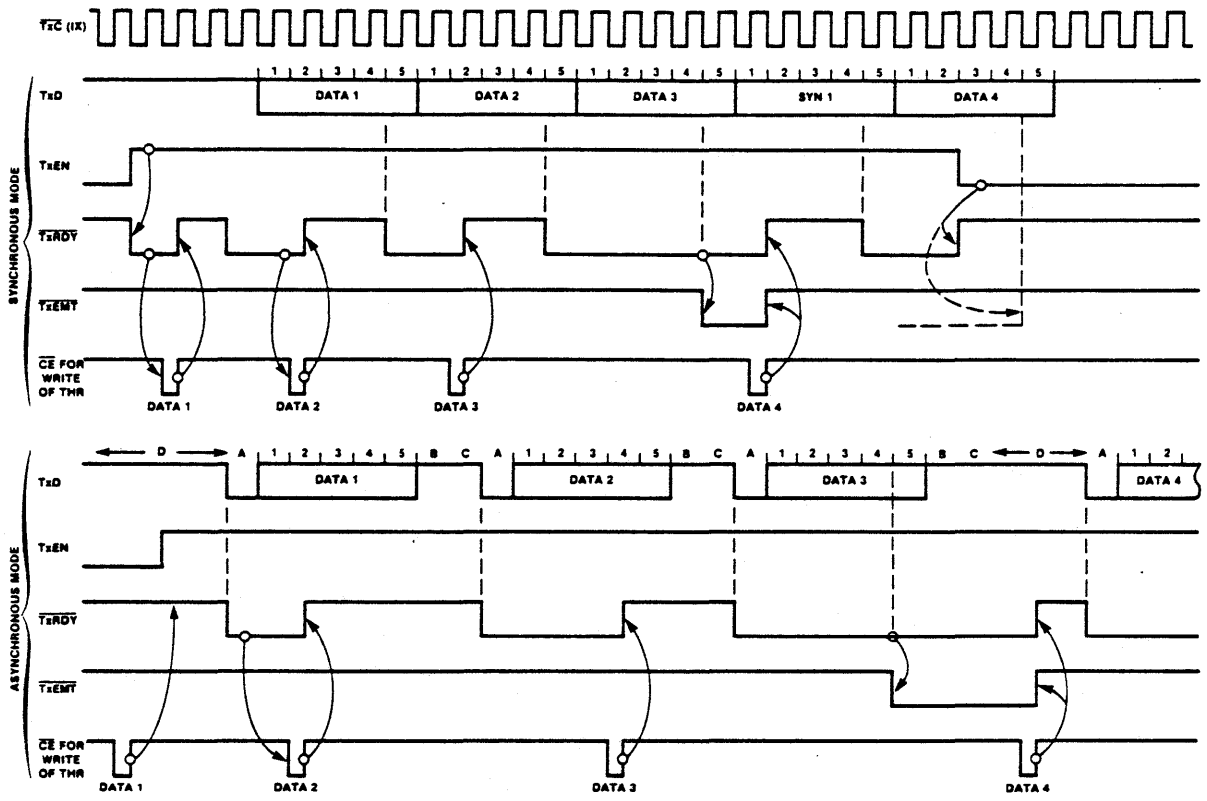


READ AND WRITE



TIMING DIAGRAMS (Cont'd)

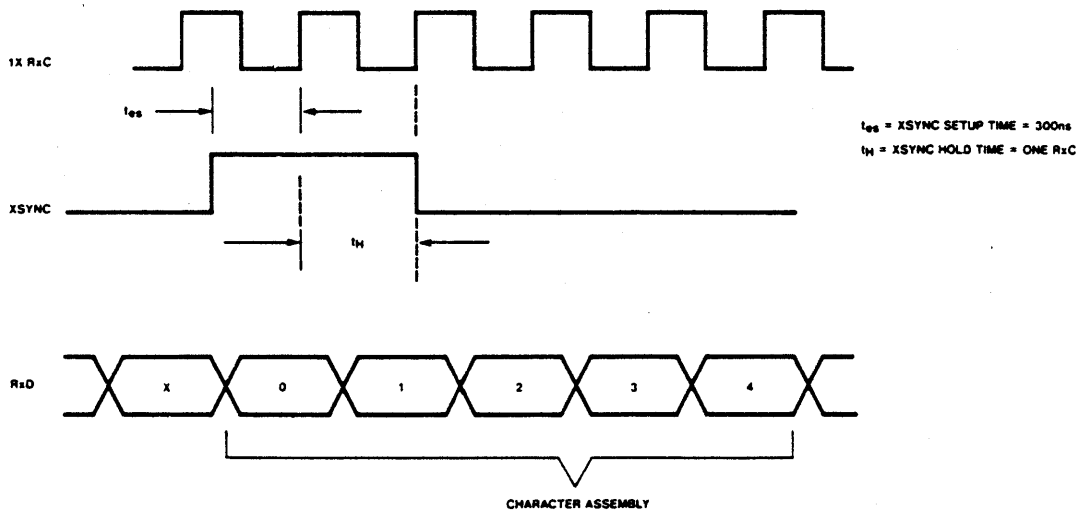
TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])



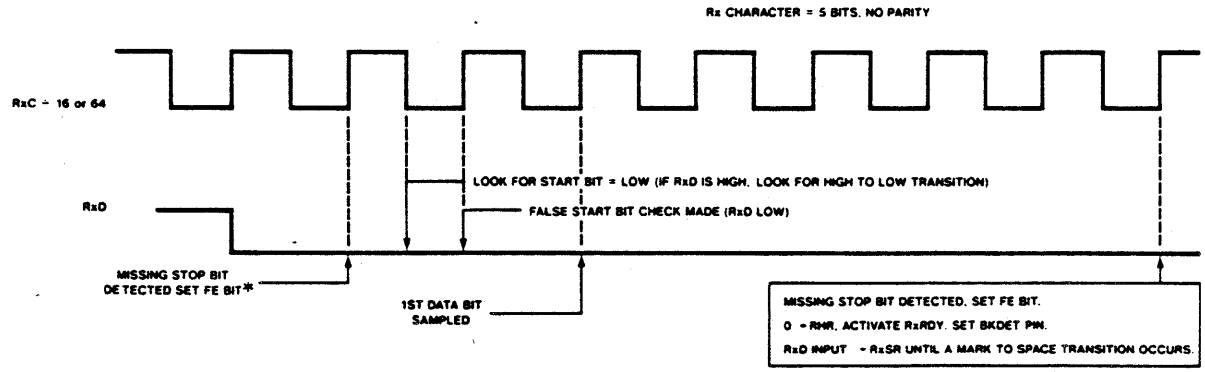
NOTES
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

TIMING DIAGRAMS (Cont'd)

EXTERNAL SYNCHRONIZATION WITH XSYNC



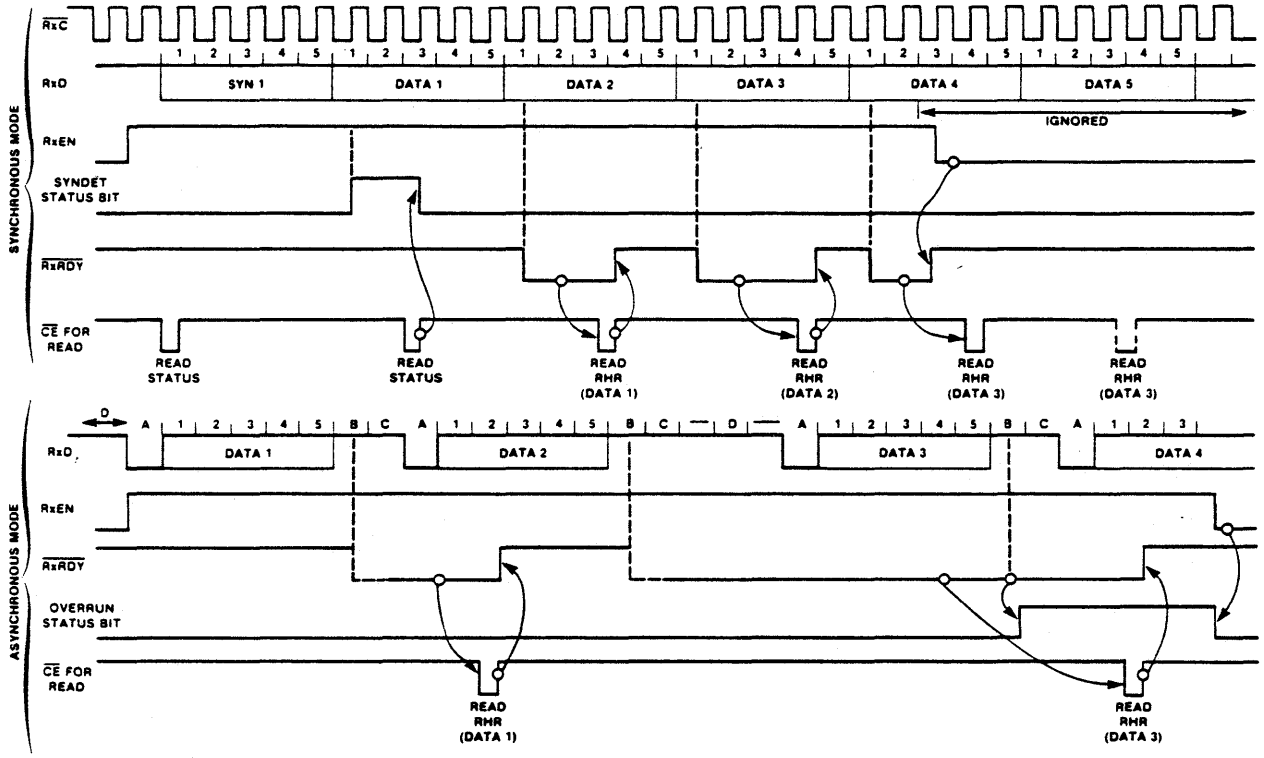
BREAK DETECTION TIMING



NOTE
* If the stop bit is present, the start bit search will commence immediately.

TIMING DIAGRAMS (Cont'd)

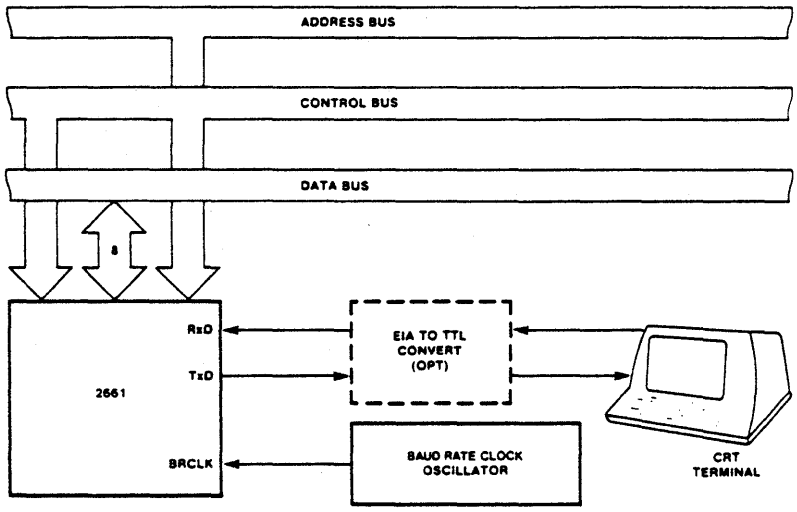
RxRDY (Shown for 5-bit characters, no parity, 2 stops bits [in asynchronous mode])



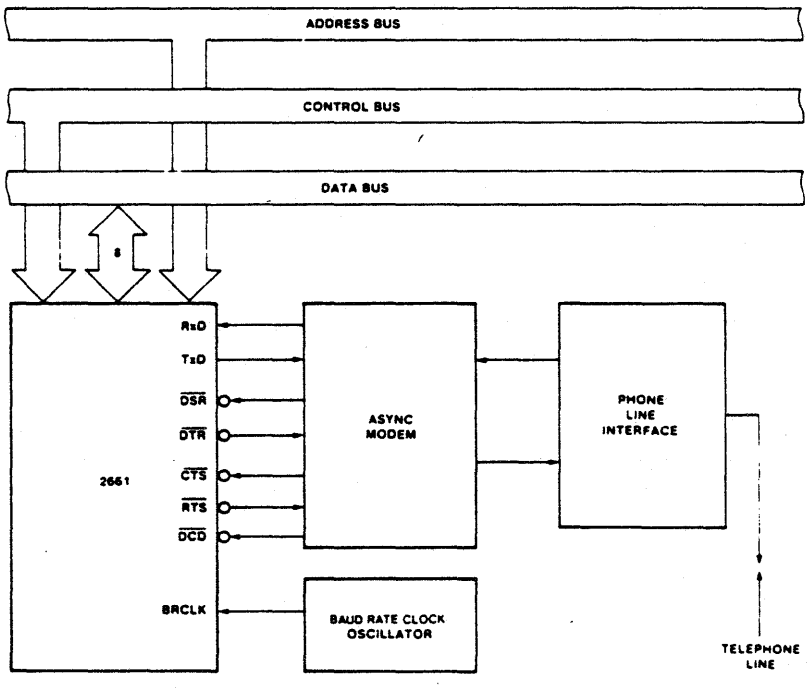
NOTES
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
Only one stop bit is detected.

TYPICAL APPLICATIONS

ASYNCHRONOUS INTERFACE TO CRT TERMINAL

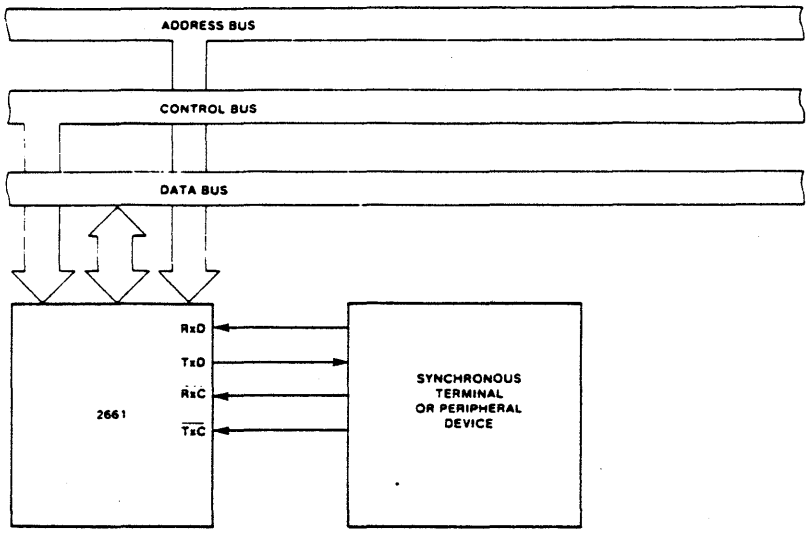


ASYNCHRONOUS INTERFACE TO TELEPHONE LINES

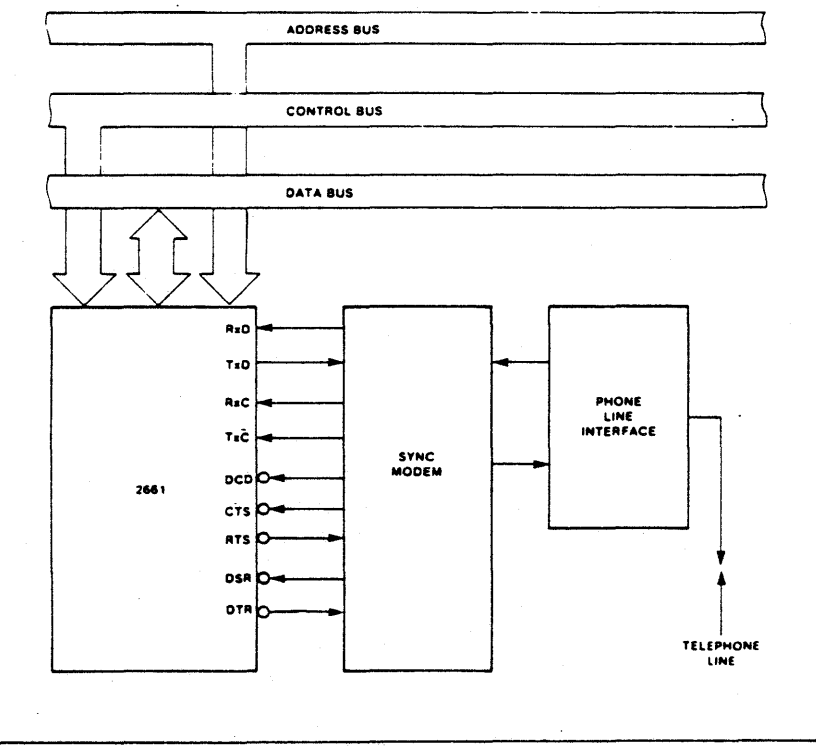


TYPICAL APPLICATIONS (Cont'd)

SYNCHRONOUS INTERFACE TO TERMINAL OR PERIPHERAL DEVICE



SYNCHRONOUS INTERFACE TO TELEPHONE LINES



signetics

a subsidiary of U.S. Philips Corporation

Signetics Corporation
811 East Arques Avenue
P.O. Box 409
Sunnyvale, California 94086
Telephone 408/739-7700

802661

© Copyright 1981 Signetics Corporation
Printed in U.S.A. 10M1181